НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для пошуку автозапчастин

Виконав студент IV курсу, групи ІТ-04
(шифр групи)

Стуканець Василь Васильович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ас., Сарнацький Владислав Віталійович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент каф. ІСТ, к.т.н., доцент Олена Жданова _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Стуканцю Василю Васильовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для пошуку автозапчастин
керівник проєкту Сарнацький Владислав Віталійович, ас.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» травня 2024 р. № 2112-с

2. Термін подання студентом проєкту «17» червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметної області, огляд ринку програмних продуктів, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання програмного забезпечення, засоби розробки, опис структури бази даних, опис архітектурного шаблону.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості розробленого програмного забезпечення, опис процесів тестування, опис контрольного прикладу.
4) Розгортання програмного забезпечення: Опис процесів розгортання розробленого програмного забезпечення, опис процесів підтримки розробленого програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	17.03.2024	
3	Постановка та формалізація задачі	23.03.2024	
4	Розробка інформаційного забезпечення	30.03.2024	
5	Алгоритмізація задачі	06.04.2024	
6	Обґрунтування вибору використаних технічних засобів	12.04.2024	
7	Розробка програмного забезпечення	20.04.2024	
8	Налагодження програми	30.04.2024	
9	Виконання графічних документів	10.05.2024	
10	Оформлення пояснювальної записки	20.05.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	17.06.2024	

Студент _____ Василь СТУКАНЕЦЬ
(підпис) (ініціали, прізвище)

Керівник _____ Владислав САРНАЦЬКИЙ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 28 таблиць, 22 рисунки та 7 джерел – загалом 62 сторінки. Дипломний проєкт присвячений розробці сайту для пошуку автозапчастин.

Метою даного дипломного проєкту є створення онлайн-платформи для покращення процесу пошуку та придбання автомобільних запчастин.

Об'єкт дослідження: процес розробки веб-застосунку для пошуку автозапчастин.

Предмет дослідження: веб-застосунок, який допомагає користувачам знайти та придбати автозапчастини онлайн.

Розділ 1: Проведено аналіз предметної області, аналіз відомих аналогів, їх переваг і недоліків. Розроблено функціональні та нефункціональні вимоги, надано можливі варіанти використання застосунку та сформульовано задачі.

Розділ 2: Змодельовано та детально описано бізнес-процеси, розглянуто особливості архітектури MVC, проаналізовано безпеку даних користувача. Детально описано інструменти для практичної реалізації застосунку та описано модулі застосунку та їх методи.

Розділ 3: Проведено тестування розробленого застосунку, описано архітектуру програмного забезпечення, вибір інструментів для розробки клієнтської та серверної частин, а також бази даних.

Розділ 4: Описано інструменти для розгортання застосунку та сам процес розгортання проаналізовано якість програмного забезпечення та описано процеси тестування.

Ключові слова: ВЕБ-ЗАСТОСУНОК, ПОШУК АВТОЗАПЧАСТИН, DJANGO, VUE.JS, MYSQL, HTML, JAVASCRIPT.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 28 tables, 22 figures and 7 sources - a total of 62 pages. The diploma project is devoted to the development of a site for finding auto parts.

The goal of this diploma project is to create an online platform to improve the process of finding and purchasing car spare parts.

Research object: the process of developing a web application for finding auto parts.

Research Subject: A web application that helps users find and purchase auto parts online.

Chapter 1: Analysis of the subject area, analysis of known analogs, their advantages and disadvantages. Functional and non-functional requirements were developed, possible options for using the application were provided, and tasks were formulated.

Chapter 2: Business processes are modeled and described in detail, features of the MVC architecture are considered, user data security is analyzed. The tools for practical implementation of the application are described in detail, and the application modules and their methods are described.

Chapter 3: The developed application was tested, the software architecture, the choice of tools for the development of the client and server parts, as well as the database, were described.

Chapter 4: Describes application deployment tools and the deployment process itself, analyzes software quality, and describes testing processes.

Keywords: WEB APPLICATION, AUTO PARTS SEARCH, DJANGO, VUE.JS, MYSQL, HTML, JAVASCRIPT.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9426.045490.01.91	Технічне завдання	11	
3	А4	КПІ.ІТ-9426.045490.02.81	Пояснювальна записка	62	
4	А4	КПІ.ІТ-9426.045490.03.13	Текст програми	44	
5	А4	КПІ.ІТ-9426.045490.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9426.045490.05.34	Керівництво користувача	12	
7	А3	КПІ.ІТ-9426.045490.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІТ-9426.045490.07.99	Схема бази даних	1	
9	А3	КПІ.ІТ-9426.045490.08.99	Креслення вигляду екранних форм	1	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ04-9426.045440.00.90		
Розроб.		Стуканець В.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Сарнацький В.В.					
Н.контр.		Радіонов П.Ю.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-04	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Вебзастосунок для пошуку автозапчастин

Технічне завдання

КПІ.ІТ04-9426.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:
_____ Павло РОДІОНОВ

Виконавець:
_____ Василь СТУКАНЕЦЬ

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ 3
2 ПІДСТАВА ДЛЯ РОЗРОБКИ 4
3 ПРИЗНАЧЕННЯ РОЗРОБКИ 5
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6
4.1 Вимоги до функціональних характеристик 6
4.1.1 Користувацького інтерфейсу 6
4.1.2 Для користувача: 6
4.2 Вимоги до надійності 6
4.3 Умови експлуатації 6
4.3.1 Вид обслуговування 7
4.3.2 Обслуговуючий персонал 7
4.4 Вимоги до складу і параметрів технічних засобів 7
4.5 Вимоги до інформаційної та програмної сумісності 7
4.5.1 Вимоги до вхідних даних 7
4.5.2 Вимоги до вихідних даних 7
4.5.3 Вимоги до мови розробки 7
4.5.4 Вимоги до середовища розробки 8
4.5.5 Вимоги до представленню вихідних кодів 8
4.6 Вимоги до маркування та пакування 8
4.7 Вимоги до транспортування та зберігання 8
4.8 Спеціальні вимоги 8
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ 9
5.1 Попередній склад програмної документації 9
5.2 Спеціальні вимоги до програмної документації 9
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ 10
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ 11

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: DeCar

Галузь застосування: Галузь застосування цієї дипломної роботи включає автомобільну промисловість, зокрема ринок автозапчастин та сервісів з технічного обслуговування і ремонту автомобілів. Розроблений сайт призначений для використання як споживачами, так і компаніями, які займаються продажем та обслуговуванням автозапчастин, з метою спрощення процесу пошуку, порівняння та придбання автомобільних запчастин через інтернет. Також платформа може бути використана в електронній комерції для поліпшення взаємодії між постачальниками автозапчастин та їх кінцевими споживачами.

Наведене технічне завдання поширюється на розробку програмного забезпечення Вебзастосунок для пошуку автозапчастин [045440], котра використовується для підбору автозапчастин.

та призначена для людей, яким необхідна допомога в підборі запчастин до власних авто.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки DeCar є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для людей, яким необхідна допомога в підборі запчастин до власних авто.

Метою розробки є створення онлайн-платформи для пошуку автозапчастин, яка надає користувачам зручний та інтуїтивно зрозумілий інтерфейс. Ця платформа повинна забезпечити ефективний механізм для швидкого пошуку та запчастин.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

– Інформативність;
– Безпека;
– Доступність;
– Швидкість реакції;
– Сучасний дизайн;
– Зручність та інтуїтивність;

4.1.2 Для користувача:

– Пошук авто за маркою;
– Пошук авто за моделлю;
– Підбір запчастин за різними категоріями;
– Перегляд інформації про запчастини та їх аналоги;

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Забезпечити належну безперебійність.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів тип процесору: Intel Core i7 – 10750h;

– об‘єм ОЗП: 32 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням будь яких операційних систем.

4.5.1 Вимоги до вхідних даних

Вимоги до вхідних даних відсутні

4.5.2 Вимоги до вихідних даних

Вимоги до вихідних даних відсутні

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python та Vue.js

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища VSCode

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді набору файлів з розширенням «py» та «vue»

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги відсутні

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема бази даних;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Вебзастосунок для пошуку автозапчастин**

КПІ.ІТ04-9426.045440.02.81

Київ – 2024

ЗМІСТ

ВСТУП 5
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7
1.1 Аналіз предметної області 7
1.2 Аналіз відомих програмних продуктів 7
1.2.1 Аналіз відомих алгоритмічних та технічних рішень 9
1.3 Опис бізнес-процесів 11
Висновки до розділу 14
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15
2.1 Варіанти використання програмного забезпечення 15
2.2 Розроблення функціональних вимог 25
2.3 Розроблення нефункціональних вимог 28
Висновки до розділу 28
3 КОНСТРУЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ... 29
3.1 Архітектура програмного забезпечення 29
3.2 Конструювання програмного забезпечення 31
3.3 Вибір інструментів для розробки клієнтської частини 33
3.4 Вибір інструментів для розробки серверної частини 35
3.5 База даних 36
Висновки до розділу 44
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧННЯ 45
4.1 Аналіз якості програмного забезпечення 45
4.2 Опис процесів тестування 45
4.3 Опис контрольного прикладу 50
Висновки до розділу 55
5 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ . 56
5.1 Розгортання програмного забезпечення 56
5.2 Вимоги до системи користувача 57
ВИСНОВКИ 59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60

ДОДАТОК А ЗВІТ ПОДІБНОСТІ..... 61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
SDK	–	Software development kit
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних.

ВСТУП

У сучасному швидкоплинному світі автомобільна промисловість відіграє вирішальну роль у забезпеченні безперебійного функціонування нашого повсякденного життя. Оскільки кількість транспортних засобів на дорогах продовжує зростати, зростає попит на надійні та ефективні послуги з технічного обслуговування та ремонту автомобілів. Важливим аспектом цього обслуговування є своєчасна та точна закупівля автозапчастин. Традиційні методи пошуку автозапчастин, які часто передбачають тривале відвідування кількох магазинів або численні телефонні запити, дедалі більше застарівають. Щоб вирішити цю проблему, розробка сайту пошуку автозапчастин пропонує багатообіцяюче рішення, яке використовує сучасні технології для спрощення процесу закупівлі запчастин.

У цій дипломній роботі досліджується дизайн і впровадження сайту пошуку автозапчастин, спрямованого на покращення в тому, як споживачі та компанії знаходять і купують автомобільні запчастини. Основною метою цього проекту є створення онлайн-платформи, яка надає користувачам комплексний і зручний інтерфейс для пошуку та порівняння та придбання автозапчастин.

Дослідження, представлене в цій дипломній роботі, ґрунтується на ретельному аналізі поточних ринкових тенденцій, потреб споживачів і технологічних досягнень у веб-розробці. Він заглиблюється в проблеми, з якими стикаються користувачі в традиційному процесі пошуку автозапчастин, і визначає ключові області, де онлайн-платформа може запропонувати значні покращення. Крім того, у цьому дослідженні описано методології, використані для розробки сайту, включаючи керування базами даних, дизайн інтерфейсу користувача та методи оптимізації пошуку.

На завершення цієї дипломнох роботи очікується, що запропонований сайт пошуку автозапчастин продемонструє потенціал значного скорочення часу та зусиль, необхідних для пошуку та придбання автомобільних

запчастин, тим самим сприяючи загальній ефективності промисловості технічного обслуговування та ремонту автомобілів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз предметної області

Автомобільна промисловість — це великий і динамічний сектор, який характеризується швидким технологічним прогресом і безперервним припливом нових моделей транспортних засобів. У цьому контексті ринок автозапчастин відіграє ключову роль у забезпеченні довговічності та продуктивності транспортних засобів. Предмет цієї дипломної роботи, розробка сайту пошуку автозапчастин, потребує комплексного аналізу кількох ключових факторів, які впливають на закупівлю та розповсюдження автозапчастин. Останніми роками ринок автозапчастин значно зріс завдяки збільшенню кількості транспортних засобів на дорогах і зростанню переваги ремонту своїми руками («зроби сам») серед власників автомобілів. Сьогодні споживачі більш схильні шукати та купувати автозапчастини в Інтернеті завдяки зручності та більш широкому вибору, який пропонують платформи електронної комерції. Ця зміна в поведінці споживачів підкреслює потребу в ефективному та зручному для користувача сайті пошуку автозапчастин онлайн. Традиційні методи пошуку автозапчастин часто передбачають відвідування кількох фізичних магазинів або зв’язок із багатьма постачальниками, що може бути трудомістким і неефективним. Ці методи ще більше ускладнюються необхідністю забезпечення сумісності між частинами та конкретною моделлю автомобіля. Відсутність централізованої бази даних і мінливість цін у різних постачальників додають складності, через що споживачам важко знайти найкращі пропозиції.

1.2 Аналіз відомих програмних продуктів

Після пошуку сайтів з пошуку запчастин було обрано 2 сайти:

– ETKA.
– MotoRad.

ETKA – це офіційний каталог для ринку Volkswagen EUROPA. Цей сайт дозволяє користувачам знаходити оригінальні запчастини Volkswagen за типом, моделлю або номером деталі [1]. Реалізовано інтерфейс користувача у вигляді веб-застосунку, інтерфейс якого зображено на рисунку 1.1

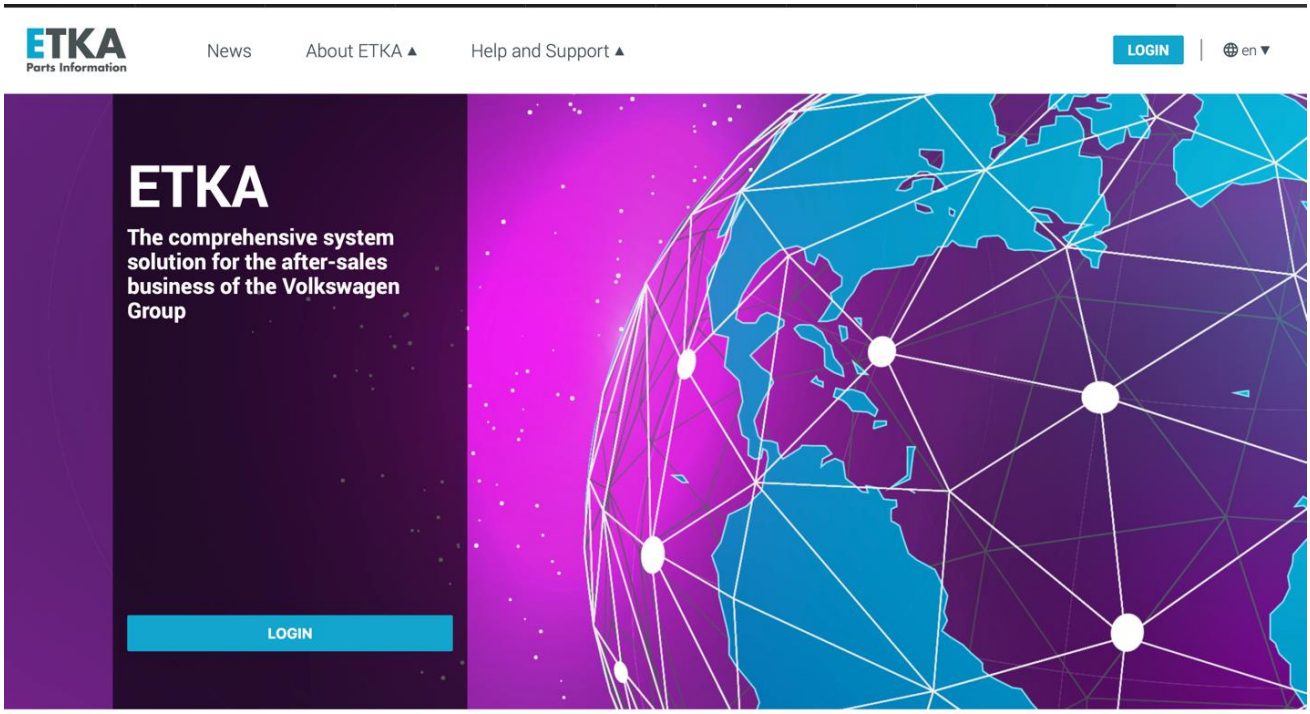

Рисунок 1.1 – Інтерфейс користувача веб-застосунку ETKA

Переваги:

– Реалізовано повний каталог запчастин по всім складовим авто.
– Даний ресурс є офіційним веб-застосунком концерну VAG.
– Зручний інтерфейс.
– Великий каталог запчастин.

Недоліки:

– Каталог містить інформацію лише про певні марки концерну VAG, а саме: Volkswagen, Skoda, Audi, Seat.
– Веб-застосунок є платним.
– Доступ до веб-застосунку мають лише офіційні дилери які співпрацюють з концерном VAG.

MotoRad [2] – розробляє, виробляє та постачає різноманітну продукцію, включаючи термостати, кришки та датчики. Ці компоненти

сумісні з існуючими та новими технологіями для двигунів внутрішнього згоряння, гібридних транспортних засобів, гібридів, що підключаються до електромережі, і акумуляторних електромобілів. Також дана компанія має власний каталог який містить інформацію про всі її продукти та їх сумісність з різними авто. Реалізовано інтерфейс користувача у вигляді веб-застосунку, інтерфейс якого зображено на рисунку 1.2

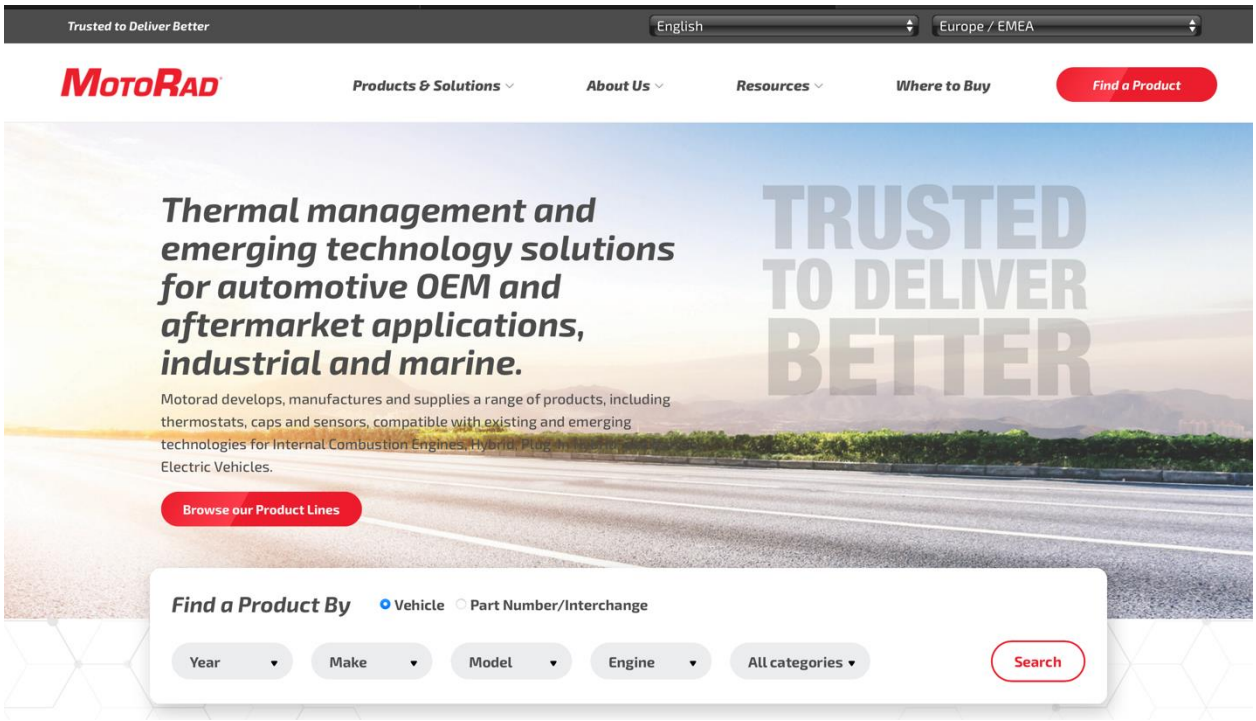

Рисунок 1.2 – Інтерфейс користувача веб-застосунку MotoRad

Переваги:

– Зручний інтерфейс.

– Великий каталог запчастин.

Недоліки:

– Список запчастин для того, чи іншого авто, включає лише ті запчастини які виробляє сама компанія.

– Мала кількість інформації про запчастини які пропонує компанія.

1.2.1 sАналіз відомих алгоритмічних та технічних рішень

У розробці «Сайту пошуку автозапчастин» ми проаналізуємо існуючу алгоритмічну підтримку та технічні рішення, які можуть полегшити

впровадження цього веб-застосунку. Цей аналіз охоплюватиме готові програмні рішення, допоміжні засоби та технології розробки.

Технічні рішення для розробки сайту пошуку автозапчастин

Django:

– Веб-фреймворк високого рівня Python, який заохочує швидку розробку та чистий, прагматичний дизайн.

– Надає вбудовані функції для автентифікації, керування базою даних і маршрутизації URL-адрес.

– Підтримується розробка RESTful API через Django REST framework (djangorestframework).

Django CORS Headers:

– Проміжне програмне забезпечення для керування спільним використанням ресурсів між джерелами (CORS), необхідне для взаємодії зовнішніх додатків із серверною частиною.

– Надає поля та утиліти для обробки телефонних номерів, забезпечуючи точне та послідовне введення даних.

– Прогресивний фреймворк JavaScript для побудови інтерфейсів користувача.

– Дозволяє розробляти інтерактивні та динамічні односторінкові програми (SPA).

– Повноцінно інтегрується з Django, створюючи розділену архітектуру інтерфейсу та серверу.

MySQL:

– Система керування реляційною базою даних, яка використовується для зберігання даних користувача, інформації про автомобіль та його деталей.

– Пропонує надійну продуктивність, масштабованість і надійність для обробки великих наборів даних.

– Підтримується такими бібліотеками, як mysql-connector-python і mysqlclient для взаємодії з базою даних.

Pillow – форк Python Imaging Library (PIL), який використовується для завдань обробки зображень, таких як зміна розміру та обрізання зображень деталей автомобіля.

Swagger – інструмент для документування та тестування API, що дозволяє автоматично генерувати документи API, спрощуючи розробку та інтеграцію клієнтських програм.

Colorama – міжплатформна утиліта для кольорового тексту терміналу, корисна для вдосконалення інструментів CLI під час розробки та налагодження.

Ці технології та інструменти разом забезпечують створення надійної, масштабованої та зручної веб-програми для пошуку та керування автозапчастинами. Використовуючи Django, Vue.js і MySQL, ми можемо створити цілісну та ефективну платформу, яка відповідає як функціональним, так і нефункціональним вимогам.

1.3 Опис бізнес-процесів

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунки 1.3 – 1.4).

Рисунок 1.3 – BPMN модель бізнес процесу «Пошук авто»

Опис бізнес процесу Пошук авто.

Ця діаграма показує бізнес-процес пошуку автомобіля, розділений на клієнтську та серверну частини.

Клієнтська частина:

– Користувач натискає на пошук: користувач натискає кнопку пошуку для початку процесу.

– Користувач намагатися вводити дані: користувач вводить необхідну інформацію для пошуку автомобіля.

Серверна частина:

– Пошук в базі даних за введеними символами: система пошуку в базі даних відповідно до введених символів користувача.

Клієнтська частина:

– Відображення запропонованих авто: система повертає результати пошуку та відображає список запропонованих автомобілів.

– Користувач вибирає авто зі списку запропонованих: користувач обирає автомобіль зі списку запропонованих варіантів.

– Перехід на наступний екран: після вибору автомобіля користувач переходить на наступний екран для подальших дій.

Цей процес показує послідовність дій користувача при пошуку автомобіля в системі з базою даних, включаючи взаємодію з інтерфейсом та обробку запиту на сервері.

Рисунок 1.4 – BPMN модель бізнес процесу «Пошук запчастин за категорією»

Ця діаграма показує бізнес-процес пошуку запчастин за категорією, розділений на клієнтську та серверну частини.

Клієнтська частина:

– Користувач обирає двигун та трансмісію: користувач вибирає необхідні параметри двигуна та трансмісії для пошуку запчастин.

Серверна частина:

– Пошук доступних двигунів та КПП: система здійснює пошук доступних двигунів та коробок передач (КПП) у базі даних.

Клієнтська частина:

– Перехід на наступний екран: користувач переходить на наступний екран після вибору двигуна та трансмісії.

– Відображення доступних категорій запчастин: система відображає категорії доступних запчастин для обраного авто.

Серверна частина:

– Пошук доступних категорій запчастин для обраного авто: система здійснює пошук доступних категорій запчастин для обраного авто на основі введених параметрів.

– Цей процес відображає послідовність дій користувача при пошуку запчастин за категорією в системі з базою даних, включаючи взаємодію з інтерфейсом та обробку запиту на сервері.

Висновки до розділу

У цьому розділі було проведено аналіз предметної області, аналіз відомих програмних продуктів та опис бізнес-процесів. Проведений аналіз дозволив сформувати базу знань для подальшої розробки програмного забезпечення. Було визначено основні вимоги до системи, які стали основою для розробки функціональних та нефункціональних вимог. Отримані результати забезпечили глибоке розуміння проблематики та необхідності розробки саме такого програмного продукту.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією даного сервісу є надання інформації, клієнту, про його автомобіль та запчастини які в ньому встановлені. Більше функцій можна побачити в таблицях 2.1 – 2.13

Таблиця 2.1 - Варіант використання UC-1

Use case name	Пошук інформації про авто
Use case ID	UC-01
Goals	Перегляд інформації про автомобіль
Actors	Користувач
Trigger	Користувач бажає переглянути доступну інформацію про автомобіль
Pre-conditions	Відкрита головна сторінка
Flow of Events	Користувач відриває головну сторінку. Користувач натискає лівою кнопкою миші на поле пошуку. Користувач починає вводити марку та модель авто. Користувач обирає свій автомобіль зі списку запропонованих авто. Користувач потрапляє на сторінку з детальною інформацією про авто.
Extension	Під час вводу тексту, система пропонує користувачу варіанти автоматичного заповнення поля пошуку
Post-Condition	Відкриття сторінки з інформацією про авто

Таблиця 2.2 - Варіант використання UC-2

Use case name	Дозволити користувачам вибрати рік випуску, двигун і трансмісію свого автомобіля.
Use case ID	UC-02
Goals	Дозволити користувачам вибрати рік випуску, двигун і трансмісію свого автомобіля.
Actors	Користувач
Trigger	Користувач вибирає модель автомобіля та перенаправляється на сторінку загальної інформації.
Pre-conditions	Користувач вибрав модель автомобіля та перебуває на сторінці загальної інформації.
Flow of Events	Користувач переглядає сторінку із загальною інформацією про обрану модель автомобіля. Користувач вибирає рік виробництва зі спадного меню. Користувач вибирає тип двигуна зі спадного меню. Користувач вибирає тип передачі зі спадного меню. Система оновлює доступні категорії деталей на основі вибраних деталей.
Extension	Якщо обрані категорії деталей недоступні, система сповістить про це користувача та запропонує альтернативні варіанти.
Post-Condition	Користувач вибрав рік, двигун і деталі трансмісії, і система відображає відповідні категорії деталей.

Таблиця 2.3 - Варіант використання UC-3

Use case name	Перегляд категорій деталей
Use case ID	UC-03
Goals	Дозволити користувачам переглядати категорії запчастин, доступних для їх автомобіля.

Продовження таблиці 2.3

Actors	Користувач
Trigger	Користувач вибрав рік випуску автомобіля, двигун і трансмісію.
Pre-conditions	Користувач вибрав дані про рік випуску автомобіля, двигун і коробку передач.
Flow of Events	Користувач переглядає сторінку з доступними категоріями деталей. Система відображає такі категорії, як масло, фільтри, гальма тощо. Користувач вибирає категорію для перегляду доступних запчастин.
Extension	Якщо немає запчастин для вибраної категорії, система сповістить користувача та запропонує відповідні категорії.
Post-Condition	Користувач перебуває на сторінці категорії деталей і готовий переглянути певні запчастини.

Таблиця 2.4 - Варіант використання UC-4

Use case name	Перегляд окремих запчастин
Use case ID	UC-04
Goals	Дозволити користувачам переглядати певні запчастини в межах вибраної категорії.
Actors	Користувач
Trigger	Користувач вибирає категорію запчастин..
Pre-conditions	Користувач вибрав категорію деталей.

Продовження таблиці 2.4

Flow of Events	Користувач переглядає сторінку з доступними запчастинами у вибраній категорії. Система відображає список запчастин, включаючи такі деталі, як ціна, марка та наявність. Користувач вибирає запчастину, щоб переглянути докладнішу інформацію.
Extension	Якщо запчастини недоступні, система повідомляє користувача та пропонує альтернативні запчастини або категорії.
Post-Condition	Користувач переглядає деталі певної запчастини.

Таблиця 2.5 - Варіант використання UC-5

Use case name	Перегляд інформації про компанію
Use case ID	UC-05
Goals	Дозволити користувачам переглянути контакнту інформацію та інформацію про компанію
Actors	Користувач
Trigger	Користувач бажає переглянути інформацію про компанію та спосіб зв’язку з нею
Pre-conditions	Користувач знаходиться на будь якій сторінці веб-застосунку
Flow of Events	Користувач натискає на кнопку «Інформація» Система перенаправляє користувача на сторінку де знаходяться дані про компанію та спосіб зв’язку з нею
Extension	-
Post-Condition	-

Таблиця 2.6 - Варіант використання UC-6

Use case name	Додати нову модель автомобіля
Use case ID	UC-06
Goals	Дозволити адміністратору додати нову модель автомобіля до бази даних.
Actors	Адміністратор
Trigger	Адміністратор хоче додати нову модель авто.
Pre-conditions	адміністратор увійшов в адмінпанель бази даних
Flow of Events	Адміністратор переходить до розділу керування моделлю автомобіля. Адміністратор натискає «Додати нову модель автомобіля». Адміністратор вводить марку, модель та інші відповідні дані. Адміністратор надсилає форму. Система зберігає нову модель автомобіля в базу даних. Система підтверджує додавання нової моделі автомобіля.
Extension	Якщо обрані категорії деталей недоступні, система сповістить про це користувача та запропонує альтернативні варіанти.
Post-Condition	Користувач вибрав рік, двигун і деталі трансмісії, і система відображає відповідні категорії деталей.

Таблиця 2.7 - Варіант використання UC-7

Use case name	Оновлення інформації про модель автомобіля
Use case ID	UC-07
Goals	дозволити адміністратору оновлювати інформацію про існуючу модель автомобіля.
Actors	Адміністратор
Trigger	адміністратор хоче оновити дані моделі автомобіля.
Pre-conditions	адміністратор увійшов в панель адміністратора і модель автомобіля існує в базі даних.

Продовження таблиці 2.7

Flow of Events	Адміністратор переходить до розділу керування моделлю автомобіля. Адміністратор вибирає модель автомобіля для оновлення. Адміністратор редагує інформацію про модель автомобіля. Адміністратор подає зміни. Система оновлює інформацію про модель автомобіля в базі даних. Система підтверджує оновлення.
Extension	-
Post-Condition	інформація про модель автомобіля оновлюється в базі даних.

Таблиця 2.8 - Варіант використання UC-8

Use case name	Додати відомості про автомобіль
Use case ID	UC-08
Goals	дозволити адміністратору додавати такі деталі, як рік, двигун і трансмісія для моделі автомобіля.
Actors	Адміністратор
Trigger	адміністратор хоче додати нові дані для моделі автомобіля.
Pre-conditions	адміністратор увійшов в панель адміністратора і модель автомобіля існує в базі даних.

Продовження таблиці 28

Flow of Events	Адміністратор переходить до розділу керування деталями автомобіля. Адміністратор вибирає модель автомобіля для додавання деталей. Адміністратор вводить рік випуску, тип двигуна та тип трансмісії. Реквізити надає адміністратор. Система зберігає нові дані в базі даних. Система підтверджує додавання нових даних.
Extension	Якщо деталі вже існують, система повідомляє адміністратора та запобігає дублюванню.
Post-Condition	До бази даних додаються нові відомості про автомобіль.

Таблиця 2.9 - Варіант використання UC-9

Use case name	Керування категоріями деталей
Use case ID	UC-09
Goals	дозволити адміністратору додавати, оновлювати або видаляти категорії деталей.
Actors	Адміністратор
Trigger	адміністратор хоче керувати категоріями деталей.
Pre-conditions	адміністратор увійшов в панель адміністратора.

Продовження таблиці 2.9

Flow of Events	Адміністратор переходить до розділу керування категоріями деталей. Адміністратор вибирає додати, оновити або видалити категорію. Адміністратор вводить або змінює деталі категорії. Адміністратор подає зміни. Система оновлює категорії деталей у базі даних. Система підтверджує дію.
Extension	У разі спроби видалити категорію, яка має пов’язані запчастини, система запитує підтвердження або запобігає видаленню.
Post-Condition	Категорії деталей керуються в базі даних.

Таблиця 2.10 - Варіант використання UC-10

Use case name	Додати нові запчастини
Use case ID	UC-10
Goals	Дозволити адміністратору додавати нові запчастини до бази даних.
Actors	Адміністратор
Trigger	Адміністратор хоче додати нові запчастини.
Pre-conditions	Адміністратор увійшов в панель адміністратора.

Продовження таблиці 2.10

Flow of Events	Адміністратор переходить до розділу керування частинами. Адміністратор натискає «Додати нову запчастину». Адміністратор вводить деталі деталей, включаючи категорію, сумісність, ціну та наявність. Адміністратор надсилає форму. Система зберігає нову деталь у базі даних. Система підтверджує додавання нової запчастини.
Extension	Якщо введені дані неповні або недійсні, система запропонує адміністратору виправити їх.
Post-Condition	нова частина додається до бази даних.

Таблиця 2.11 - Варіант використання UC-11

Use case name	Оновити інформацію про запчастину
Use case ID	UC-11
Goals	дозволити адміністратору оновлювати інформацію про наявну запчастину.
Actors	Адміністратор
Trigger	адміністратор хоче оновити деталі деталі..
Pre-conditions	адміністратор увійшов в адмінпанель і частина існує в базі даних.
Flow of Events	Адміністратор переходить до розділу керування частинами. Адміністратор вибирає запчастину для оновлення. Адміністратор редагує інформацію про запчастину. Адміністратор подає зміни. Система оновлює інформацію про запчастини в базі даних. Система підтверджує оновлення.
Extension	-

Продовження таблиці 2.11

Post-Condition	інформація про запчастину оновлюється в базі даних.

Таблиця 2.12 - Варіант використання UC-12

Use case name	видалити запчастину
Use case ID	UC-08
Goals	дозволити адміністратору видалити запчастину з бази даних.
Actors	Адміністратор
Trigger	адміністратор хоче видалити запчастину.
Pre-conditions	адміністратор увійшов в адмінпанель і частина існує в базі даних.
Flow of Events	Адміністратор переходить до розділу керування частинами. Адміністратор вибирає запчастину для видалення. Адміністратор підтверджує видалення. Система видаляє запчастину з бази даних. Система підтверджує видалення.
Extension	-
Post-Condition	деталь видалено з бази даних.

Таблиця 2.13 - Варіант використання UC-13

Use case name	Перегляд системних журналів
Use case ID	UC-13
Goals	дозволити адміністратору переглядати системні журнали для моніторингу та усунення несправностей.
Actors	Адміністратор
Trigger	адміністратор увійшов в панель адміністратора.

Продовження таблиці 2.13

Pre-conditions	адміністратор увійшов в панель адміністратора і модель автомобіля існує в базі даних.
Flow of Events	Адміністратор переходить до розділу системних журналів. Адміністратор вибирає потрібний файл журналу або діапазон дат. Система відображає вибрані журнали. Адміністратор переглядає журнали з метою моніторингу або усунення несправностей..
Extension	-
Post-Condition	адміністратор переглядає та аналізує системні журнали.

2.2 Розроблення функціональних вимог

Програмне забезпечення поділено на модулі, кожен з яких має свій власний набір функцій. Таблиця 2.14 представляє загальну модель вимог, а таблиця 2.15 містить опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог зображено на рисунку 2.1.

Таблиця 2.14 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1	Пошук інформації про автомобіль	FR-01	Високий	Середній
2	Вибір деталей автомобіля	FR-02	Високий	Високий
3	Перегляд категорій деталей	FR-03	Високий	Високий
4	Перегляд конкретних запчастин	FR-04	Середній	Низький

Продовження таблиці 2.14

5	Перегляд інформації про компанію	FR-05	Високий	Середній
6	Додати нову модель автомобіля	FR-06	Високий	Середній
7	Оновити інформацію про модель автомобіля	FR-07	Середній	Низький
8	Додати відомості про автомобіль	FR-08	Високий	Середній
9	Керування категоріями деталей	FR-09	Високий	Середній
10	Додавання нових запчастин	FR-10	Середній	Низький
11	Оновити інформацію про запчастини	FR-11	Низький	Низький
12	Видалити запчастину	FR-12	Низький	Низький
13	Перегляд системних журналів	FR-13	Високий	Високий

Таблиця 2.15 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-01	Пошук інформації про автомобіль: система повинна дозволяти користувачам шукати інформацію про автомобіль за маркою та моделлю, надаючи список відповідних автомобілів і перенаправляючи на сторінку з детальною інформацією після вибору.
FR-02	Вибір деталей автомобіля: система повинна дозволяти користувачам вибирати рік випуску, двигун і деталі трансмісії для свого автомобіля, а також оновлювати доступні категорії деталей на основі цих виборів.
FR-03	Перегляд категорій деталей: система повинна відображати категорії деталей, доступних для вибраних деталей автомобіля, наприклад масло, фільтри та гальма.

Продовження таблиці 2.16

FR-04	Перегляд конкретних запчастин: система повинна дозволяти користувачам переглядати конкретні запчастини в межах вибраної категорії, включаючи такі деталі, як ціна, марка та наявність.
FR-05	Перегляд інформації про компанію: система повинна надати сторінку з інформацією про компанію та контактними даними, доступну з будь-якої сторінки веб-застосунку.
FR-06	Додати нову модель автомобіля: система повинна дозволяти адміністратору додавати нову модель автомобіля до бази даних, включаючи марку, модель та інші відповідні деталі.
FR-07	Оновити інформацію про модель автомобіля: система повинна дозволяти адміністратору оновлювати існуючу інформацію про модель автомобіля в базі даних.
FR-08	Додати відомості про автомобіль: система повинна дозволяти адміністратору додавати такі відомості, як рік, двигун і трансмісія для моделі автомобіля.
FR-09	Керування категоріями деталей: система повинна дозволяти адміністратору додавати, оновлювати або видаляти категорії деталей у базі даних.
FR-10	Додавання нових запчастин: система повинна дозволяти адміністратору додавати нові запчастини до бази даних, включаючи категорію, сумісність, ціну та доступність.
FR-11	Оновити інформацію про запчастини: система має дозволити адміністратору оновлювати існуючу інформацію про деталі в базі даних.
FR-12	Видалити запчастину: система повинна дозволити адміністратору видаляти запчастини з бази даних.

Продовження таблиці 2.16

FR-13	Перегляд системних журналів: система повинна дозволяти адміністратору переглядати системні журнали для моніторингу та усунення несправностей, надаючи опції для вибору файлів журналів або діапазонів дат.

2.3 Розроблення нефункціональних вимог

Для веб-застосунку було висунуто наступні нефункціональні вимоги:

– Система має бути масштабованою, щоб відповідати майбутньому зростанню як бази користувачів, так і обсягу даних, не вимагаючи серйозної зміни архітектури.

– Система повинна мати інтуїтивно зрозумілий і зручний інтерфейс із зрозумілою навігацією та функціями пошуку, що гарантує, що користувачі зможуть легко знайти потрібну інформацію та запчастини без тривалого навчання чи підтримки.

– Система має бути сумісною з основними веб-браузерами (Chrome, Firefox, Safari, Edge).

– Система повинна підтримувати українську мову.

Висновки до розділу

У цьому розділі було розроблено варіанти використання програмного забезпечення, функціональні та нефункціональні вимоги. Було визначено основні сценарії взаємодії користувачів із системою, що дозволило чітко окреслити функціональні можливості програмного продукту. Розроблені вимоги стали основою для подальшого конструювання та аналізу програмного забезпечення, а також забезпечили чітке розуміння необхідних функцій та характеристик системи.

3 КОНСТРУЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для розробки сайту для пошуку автомобільних запчастин було використано відразу декілька архітектурних патернів. Для серверної частини застосунку було обрано шаблон MVC — це основний шаблон проектування програмного забезпечення, який використовується для поділу програми на три взаємопов’язані компоненти: модель, view та контролер. Такий поділ допомагає керувати складними програмами, розділяючи логіку програми на окремі секції, що відповідають за різні аспекти.

Модель: цей компонент обробляє дані та бізнес-логіку. Він відповідає за пошук даних, зберігання та будь-які бізнес-правила чи логіку, застосовані до даних. У Django моделі — це класи Python, які визначають структуру таблиць і зв’язків бази даних.

View: компонент перегляду керує відображенням інформації. Він отримує дані від моделі та відображає їх користувачеві. В даній дипломній роботі компонент view — це функції або класи, які отримують веб-запити, витягують дані з моделі та відображають веб-сторінку за допомогою шаблонів.

Контролер: контролер діє як посередник між моделлю та view. Він обробляє введені користувачем дані, оновлює модель і повертає вихідний дисплей користувачеві. У Django диспетчер URL-адрес (urls.py) і функції view спільно діють як контролер, направляючи запити користувачів до відповідної функції перегляду.

Переваги:

– Розподіл завдань: розподіляючи відповідальність, MVC покращує читабельність коду та зручність обслуговування.

– Масштабованість: дозволяє паралельну розробку, оскільки різні розробники можуть незалежно працювати над моделлю, представленням і контролером.

– Повторне використання: компоненти можна повторно використовувати в різних частинах програми.

Для клієнтської частини веб-застосунку було обрано компонентну архітектуру. Компонентна архітектура передбачає декомпозицію програмної системи на незалежні багаторазові частини, відомі як компоненти. Кожен компонент інкапсулює свою функціональність, що полегшує керування та розробку.

Vue.js реалізує архітектуру на основі компонентів, де додаток створено за допомогою ієрархії компонентів. Кожен компонент має власний шаблон, стиль і логіку, інкапсульовані в одному файлі. Також кожен компонент є автономним і може розроблятися, тестуватися та підтримуватися незалежно. Ця інкапсуляція спрощує налагодження та сприяє багаторазовому використанню компонента в застосунку. Також варто зазначити, що компоненти можуть спілкуватися один з одним за допомогою реквізитів і подій. Реквізити використовуються для передачі даних від батьківських до дочірніх компонентів, тоді як події використовуються для надсилання повідомлень від дочірніх до батьківських компонентів.

Переваги:

– Модульність: програма розбита на менші керовані частини.

– Повторне використання: компоненти можна повторно використовувати в різних частинах програми, зменшуючи дублювання коду.

– Ремонтопридатність: зміни в одному компоненті мінімально впливають на інші компоненти, спрощуючи технічне обслуговування та оновлення.

Загалом застосункок було роброблено за допомогою клієнт-серверної архітектури. Клієнт-серверна архітектура — це розподілена структура додатків, яка розподіляє завдання між серверами та клієнтами. Сервер

розміщує, надає та керує більшістю ресурсів і послуг, тоді як клієнт отримує доступ до цих послуг і взаємодіє з ними.

Клієнт (інтерфейс): інтерфейс створено з використанням Vue.js, що забезпечує користувацький інтерфейс і взаємодію з користувачем. Він надсилає запити на серверний сервер і обробляє відповіді для динамічного оновлення інтерфейсу користувача.

Сервер (бекенд): бекенд, розроблений за допомогою Python спільно з Django, він забезпечує обробку даних, бізнес-логіку та взаємодію з базою даних. Він надає RESTful API, які інтерфейс може викликати для виконання таких операцій, як отримання, оновлення або видалення даних.

Зв’язок: клієнт і сервер спілкуються через HTTP/HTTPS за допомогою RESTful API. Кожен запит клієнта не має стану, тобто містить усю інформацію, необхідну серверу для виконання запиту.

Переваги:

– Масштабованість: Клієнт і сервер можна масштабувати незалежно залежно від навантаження.

– Гнучкість: клієнт і сервер можна розробляти за допомогою різних технологій і розвивати окремо.

3.2 Конструювання програмного забезпечення

Далі наведено детальний опис основних класів.

Ключовою задачею класу CarModel є надання інформації про моделі автомобілів, а саме, марка, модель і рік.

Список методів та їх опис наведено в таблиці 3.1

Таблиця 3.1 – Короткий опис класу CarModel

__str__(self)	Повертає String для представлення моделі автомобіля.
get_absolute_url(self)	Повертає URL для детального перегляду інформації про модель автомобіля.

Ключовою задачею класу PartCategory є відображення списку доступних категорій запчастин. Список методів та їх опис наведено в таблиці 3.2

Таблиця 3.2 – Короткий опис класу PartCategory

__str__(self)	Повертає String для відображення категорії запчастини

Ключовою задачею класу Part є відображення інформації про запчастину. Список методів та їх опис наведено в таблиці 3.3

Таблиця 3.3 – Короткий опис класу Part

__str__(self)	Повертає String для відображення категорії запчастини

CarModelView обробляє операції CRUD для моделей автомобілів. Список методів та їх опис наведено в таблиці 3.4

Таблиця 3.4 – Короткий опис класу CarModelView

get(self, request)	Отримує моделі автомобілів.
post(self, request)	створює нову модель автомобіля.
put(self, request, pk)	оновлює існуючу модель автомобіля.
delete(self, request, pk)	видаляє модель автомобіля

PartView обробляє операції CRUD для запчастин для автомобілів. Список методів та їх опис наведено в таблиці 3.5

Таблиця 3.5 – Короткий опис класу PartView

get(self, request)	Отримує моделі автомобілів.
post(self, request)	створює нову модель автомобіля.
put(self, request, pk)	оновлює існуючу модель автомобіля.
delete(self, request, pk)	видаляє модель автомобіля

SearchView Забезпечує пошук моделей і запчастин автомобілів.

Таблиця 3.7 – Короткий опис класу SearchView Список методів та їх опис наведено в таблиці 3.6

get(self, request)	Виконує пошуковий запит.
post(self, request)	Обробляє складні пошукові запити.

SearchComponent надає інтерфейс користувача для пошуку моделей автомобілів і запчастин.

Таблиця 3.7 – Короткий опис класу SearchComponent

searchCarModel()	Пошук моделей автомобілів.
searchPart()	Шукає запчастин

CarInfoComponent відображає детальну інформацію про обрану модель автомобіля.

Таблиця 3.8 – Короткий опис класу CarInfoComponent

fetchCarDetails()	Отримує детальну інформацію про модель автомобіля.

PartCategoryComponent надає перелік доступних категорій запчастин для вибраної моделі автомобіля.

Таблиця 3.9 – Короткий опис класу PartCategoryComponent

etchCategories()	Отримує категорії деталей.

Shows details of parts within a selected category.

Таблиця 3.10 – Короткий опис класу PartDetailsComponent

fetchPartDetails()	Отримує деталі частин у категорії.

3.3 Вибір інструментів для розробки клієнтської частини

Для написання клієнської частини веб-застосунку було обрано JavaScript та фреймворки які доступні для нього. JavaScript — це універсальна мова програмування, необхідна для веб-розробки[3]. У цьому проекті він працює разом із Vue.js для керування логікою програми та взаємодією з користувачем. JavaScript дозволяє динамічно оновлювати вміст і інтерактивні елементи без перезавантаження сторінки, покращуючи

взаємодію з користувачем. Маніпулюючи DOM, JavaScript дозволяє розробникам створювати адаптивні інтерфейси, які реагують на введення користувачами в режимі реального часу. Це також полегшує асинхронні операції, такі як виклики API, забезпечуючи плавну та ефективну обробку даних.

Vue.js[4] — це прогресивний фреймворк, розроблений для побудови інтерфейсів користувача. Він надає доступну основну бібліотеку, яка зосереджена виключно на рівні view, що дозволяє легко підібрати та інтегрувати з іншими проектами. Vue.js використовує архітектуру на основі компонентів, що дозволяє розробникам створювати інкапсульовані компоненти, які можна використовувати повторно, які керують власним станом і поведінкою. Ця модульність полегшує масштабованість у складних програмах.

Система реагування Vue забезпечує ефективне оновлення DOM, відстежуючи залежності та запускаючи оновлення лише за необхідності. HTML (HyperText Markup Language) є основною мовою для створення веб-сторінок. Він структурує вміст програми, визначаючи такі елементи, як заголовки, абзаци, посилання та зображення. HTML забезпечує семантичну структуру, необхідну для доступності та оптимізації пошукових систем. У цьому проекті шаблони HTML у компонентах Vue визначають макет і структуру кожної частини інтерфейсу користувача, забезпечуючи чітке та організоване представлення вмісту.

У великих програмах керування станом кількох компонентів може стати складним завданням. Vue.js вирішує цю проблему за допомогою Vuex, бібліотеки керування станом, розробленої спеціально для програм Vue. Vuex централізує керування станом, дозволяючи всім компонентам спільно використовувати та реагувати на той самий стан, забезпечуючи послідовність і передбачуваність у всій програмі. Vue.js має сильну спільноту та багату екосистему бібліотек та інструментів. Це включає Vue Router для керування навігацією додатків, Vuex для керування станом, а також широкий спектр

сторонніх компонентів і плагінів, які розширюють функціональні можливості додатків Vue. Активна спільнота забезпечує постійні вдосконалення та підтримку, що робить Vue.js надійним вибором для інтерфейсної розробки.

Поєднання Vue.js, JavaScript і HTML створює надійну інтерфейсну архітектуру. Vue.js керує структурою стану та компонентів, JavaScript обробляє логіку програми та інтерактивність, а HTML визначає структуру вмісту. Ця інтеграція забезпечує плавний робочий процес розробки, коли кожен компонент можна розробляти, тестувати та підтримувати незалежно перед інтеграцією.

Використання однофайлових компонентів Vue (SFC) ще більше покращує цей процес, інкапсулюючи HTML, JavaScript і CSS для кожного компонента в одному файлі, сприяючи кращій організації та повторному використанню. Використання Vue.js сприяє декларативному підходу програмування, коли розробники описують бажаний стан інтерфейсу користувача, а Vue піклується про оновлення DOM відповідно до цього стану. Це контрастує з імперативним підходом, коли розробники вручну маніпулюють DOM. Декларативність Vue.js спрощує процес розробки та знижує ймовірність помилок.

3.4 Вибір інструментів для розробки серверної частини

Для написання серверної частини застосунку було обрано мову програмування Python. Python[5] — це інтерпретована мова програмування високого рівня, відома своєю читабельністю, простотою та універсальністю. Python став однією з найпопулярніших мов програмування в усьому світі завдяки легкості вивчення та широкому застосуванню в різних сферах.

Django — це високорівневий веб-фреймворк Python, який допомагає швидкій розробці. Він надає широкий спектр вбудованих функцій, включаючи систему ORM (Object-Relational Mapping), механізми автентифікації та інтерфейс адміністратора[6]. Філософія Django визначає, що він поставляється з великою кількістю вбудованих функцій для

вирішення поширених завдань веб-розробки, що дозволяє розробникам зосередитися на написанні логіки своїх програм, а не винаходити велосипед заново.

Django REST Framework[7] — це потужний інструментарій для створення веб-API. Він легко інтегрується з Django ORM і надає такі широкі можливості, як серіалізатори для перетворення складних типів даних у рідні типи даних Python, набори view для організації логіки ендпоїнтів і широкий спектр класів дозволів. DRF спрощує процес створення надійних, масштабованих RESTful API за допомогою Django.

Django CORS Headers — це програма, яка обробляє серверні заголовки, необхідні для спільного використання ресурсів між джерелами (CORS). Це дозволяє вашій серверній частині безпечно спілкуватися з інтерфейсними програмами, розміщеними на різних джерелах, керуючи політикою CORS. Це має вирішальне значення для того, щоб веб-застосунки могли надсилати запити API до вашої серверної частини з різних доменів.

SQLParse — це бібліотека для аналізу операторів SQL. Він надає інструменти для розділення, форматування та аналізу запитів SQL. У Django SQLParse використовується внутрішньо для обробки та форматування запитів SQL, що полегшує керування та налагодження взаємодії з базою даних.

3.5 База даних

У якості бази даних, при розробці веб-застосунку, було обрано PostgreSQL. Схему бази даних зображено на рисунку 3.1

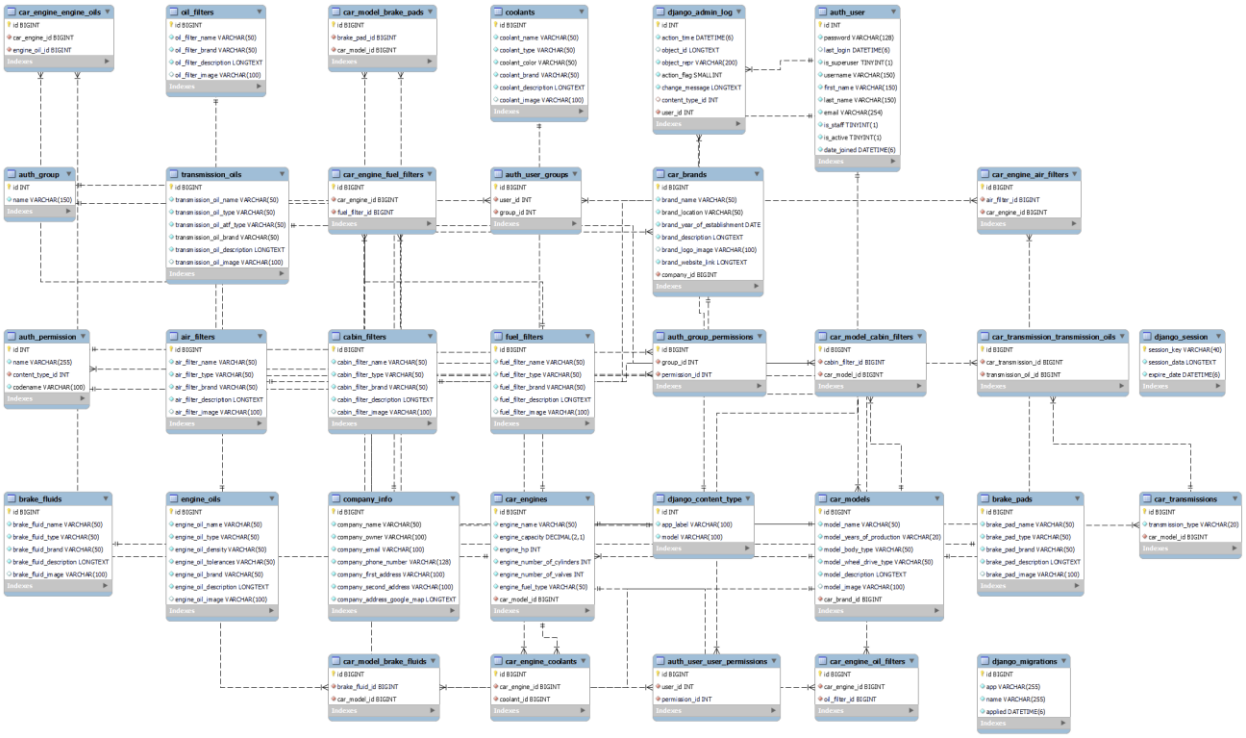

Рисунок 3.1 – Схема бази даних

Детальний опис бази даних наведено в таблицях 3.11 – 3.40

Таблиця 3.11 – Опис таблиці car_engine_engine_oils

Атрибут	**Тип**	**Опис**
Id	INT	Унікальний ідентифікатор
car_engine_id	INT	Ідентифікатор двигуна автомобіля
engine_oil_id	INT	Ідентифікатор моторної оливи

Таблиця 3.12 – Опис таблиці air_filters

Атрибут	**Тип**	**Опис**
Id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
filter_name	VARCHAR	Назва фільтра
part_number	VARCHAR	Номер частини
price	DECIMAL	Ціна
compatibility	VARCHAR	Сумісність

Таблиця 3.13 – Опис таблиці transmission_oils

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_transmission_id	INT	Ідентифікатор трансмісії автомобіля
transmission_oil_id	INT	Ідентифікатор трансмісійної оливи

Таблиця 3.14 – Опис таблиці car_model_brake_pads

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
brake_pad_id	INT	Ідентифікатор гальмівних колодок

Таблиця 3.15 – Опис таблиці coolants

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
coolant_name	VARCHAR	Назва охолоджувальної рідини
part_number	VARCHAR	Номер частини
price	DECIMAL	Ціна
compatibility	VARCHAR	Сумісність

Таблиця 3.16 – Опис таблиці auth_user_groups

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
user_id	INT	Ідентифікатор користувача
group_id	INT	Ідентифікатор групи

Таблиця 3.17 – Опис таблиці auth_permission

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
name	VARCHAR	Назва дозволу
content_type_id	INT	Ідентифікатор типу контенту
codename	VARCHAR	Кодове ім'я

Таблиця 3.18 – Опис таблиці fuel_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
filter_name	VARCHAR	Назва фільтра
part_number	VARCHAR	Номер частини
price	DECIMAL	Ціна
compatibility	VARCHAR	Сумісність

Таблиця 3.19 – Опис таблиці cabin_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
filter_name	VARCHAR	Назва фільтра
part_number	VARCHAR	Номер частини
price	DECIMAL	Ціна
compatibility	VARCHAR	Сумісність

Таблиця 3.20 – Опис таблиці auth_group_permissions

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
group_id	INT	Ідентифікатор групи
permission_id	INT	Ідентифікатор дозволу

Таблиця 3.21 – Опис таблиці car_model_cabin_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
cabin_filter_id	INT	Ідентифікатор фільтра салону

Таблиця 3.22 – Опис таблиці car_transmission_transmission_oils

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_transmission_id	INT	Ідентифікатор трансмісії автомобіля
transmission_oil_id	INT	Ідентифікатор трансмісійної оливи

Таблиця 3.23 – Опис таблиці auth_user_user_permissions

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
user_id	INT	Ідентифікатор користувача
permission_id	INT	Ідентифікатор дозволу

Таблиця 3.24 – Опис таблиці django_admin_log

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
action_time	DATETIME	Час дії
user_id	INT	Ідентифікатор користувача
content_type_id	INT	Ідентифікатор типу контенту
object_id	TEXT	Ідентифікатор об'єкта
object_repr	VARCHAR	Представлення об'єкта
action_flag	SMALLINT	Прапор дії
change_message	TEXT	Повідомлення про зміну

Таблиця 3.25 – Опис таблиці auth_group

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
name	VARCHAR	Назва групи

Таблиця 3.26 – Опис таблиці car_engine_air_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_engine_id	INT	Ідентифікатор двигуна автомобіля
air_filter_id	INT	Ідентифікатор повітряного фільтра

Таблиця 3.27 – Опис таблиці auth_user

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
password	VARCHAR	Пароль
last_login	DATETIME	Останній вхід
is_superuser	BOOLEAN	Чи є суперкористувачем
username	VARCHAR	Ім'я користувача
first_name	VARCHAR	Ім'я
last_name	VARCHAR	Прізвище
email	VARCHAR	Електронна пошта
is_staff	BOOLEAN	Чи є персоналом
is_active	BOOLEAN	Чи активний
date_joined	DATETIME	Дата приєднання

Таблиця 3.28 – Опис таблиці django_content_type

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
app_label	VARCHAR	Мітка застосунку
model	VARCHAR	Модель

Таблиця 3.29 – Опис таблиці car_models

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
name	VARCHAR	Назва моделі
manufacturer	VARCHAR	Виробник
year	INT	Рік випуску

Таблиця 3.30 – Опис таблиці brake_pads

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
brake_pad_name	VARCHAR	Назва гальмівної колодки
part_number	VARCHAR	Номер частини

Продовження таблиці 3.30

price	DECIMAL	Ціна
compatibility	VARCHAR	Сумісність

Таблиця 3.31 – Опис таблиці car_permissions

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
name	VARCHAR	Назва дозволу
content_type_id	INT	Ідентифікатор типу контенту
codename	VARCHAR	Кодове ім'я

Таблиця 3.32 – Опис таблиці engine_oils

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
engine_oil_name	VARCHAR	Назва моторної оливи
part_number	VARCHAR	Номер частини
price	DECIMAL	Ціна
compatibility	VARCHAR	Сумісність

Таблиця 3.33 – Опис таблиці company_info

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
company_name	VARCHAR	Назва компанії
address	VARCHAR	Адреса
phone	VARCHAR	Телефон
email	VARCHAR	Електронна пошта
website	VARCHAR	Вебсайт

Таблиця 3.34 – Опис таблиці django_migrations

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
app	VARCHAR	Застосунок
name	VARCHAR	Назва міграції

Продовження таблиці 3.34

applied	DATETIME	Дата застосування

Таблиця 3.35 – Опис таблиці django_session

Атрибут	**Тип**	**Опис**
session_key	CHAR	Ключ сесії
session_data	TEXT	Дані сесії
expire_date	DATETIME	Дата завершення сесії

Таблиця 3.36 – Опис таблиці car_model_oil_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
oil_filter_id	INT	Ідентифікатор масляного фільтра

Таблиця 3.37 – Опис таблиці car_model_fuel_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
fuel_filter_id	INT	Ідентифікатор паливного фільтра

Таблиця 3.38 – Опис таблиці car_model_air_filters

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
air_filter_id	INT	Ідентифікатор повітряного фільтра

Таблиця 3.39 – Опис таблиці car_model_brake_fluid

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
brake_fluid_id	INT	Ідентифікатор гальмівної рідини

Таблиця 3.40 – Опис таблиці car_model_coolants

Атрибут	**Тип**	**Опис**
id	INT	Унікальний ідентифікатор
car_model_id	INT	Ідентифікатор моделі автомобіля
coolant_id	INT	Ідентифікатор охолоджувальної рідини

Висновки до розділу

У цьому розділі було описано архітектуру програмного забезпечення, процес конструювання, вибір інструментів для розробки клієнтської та серверної частини, а також розробку бази даних. Було визначено основні компоненти системи та їх взаємодію, що забезпечило логічну та структуровану організацію програмного продукту. Вибір інструментів та технологій дозволив забезпечити ефективну розробку та подальше обслуговування системи.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧННЯ

4.1 Аналіз якості програмного забезпечення

Для аналізу якості програмного забезпечення було використано веб-застосунок embold.io.

Загальну оцінку якості, та звіт можна побачити на рисунку 4.1

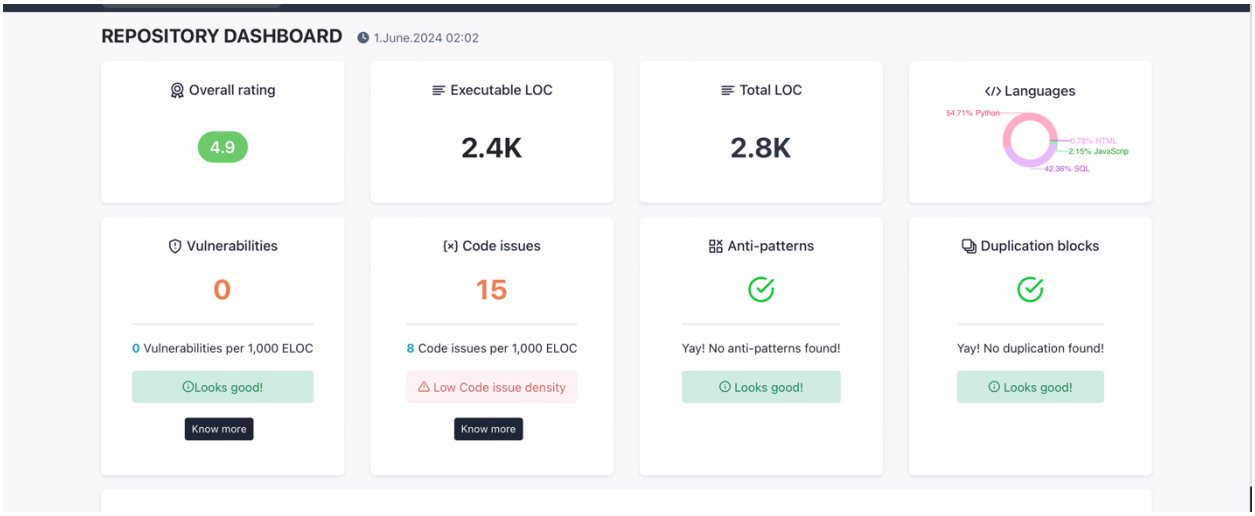

Рисунок 4.1 – Звіт та оцінка ПЗ

4.2 Опис процесів тестування

Для тестування функціональної частини програмного забезпечення було обрано мануальний спосіб. Опис проведених тестів наведено нижче в таблицях 4.1 – 4.8

Таблиця 4.1 – Тест 1.1. Перевірка пошуку автомобіля за моделлю

Тест	Перевірка пошуку автомобіля за моделлю
Модуль	Пошук автомобіля
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на головній сторінці сайту
Вхідні дані	Назва моделі автомобіля "Octavia"

Продовження таблиці 4.1

Опис проведення тесту	Користувач вводить назву моделі автомобіля в поле пошуку
Очікуваний результат	Система успішно проводить пошук моделі в базі даних та пропонує користувачу варіанти
Фактичний результат	Система успішно проводить пошук моделі в базі даних та пропонує користувачу варіанти

Таблиця 4.2 – Тест 1.2. Перевірка пошуку автомобіля за маркою

Тест	Перевірка пошуку автомобіля за маркою
Модуль	Пошук автомобіля
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на головній сторінці сайту
Вхідні дані	Назва марки автомобіля "Volkswagen"
Опис проведення тесту	Користувач обирає марку автомобіля зі списку
Очікуваний результат	Система успішно відображає всі доступні моделі авто марки Volkswagen
Фактичний результат	Система успішно відображає всі доступні моделі авто марки Volkswagen

Таблиця 4.3 – Тест 1.3. Перевірка вибору автомобіля за моделлю

Тест	Перевірка вибору автомобіля за моделлю
Модуль	Вибір автомобіля
Номер тесту	1.3

Продовження таблиці 4.3

Початковий стан системи	Користувач знаходиться на сторінці з доступними моделями автомобілів марки Volkswagen
Вхідні дані	Модель автомобіля "Golf"
Опис проведення тесту	Користувач обирає модель автомобіля зі списку
Очікуваний результат	Система переведе користувача на сторінку автомобіля та відобразить основну інформацію про нього
Фактичний результат	Система перевела користувача на сторінку автомобіля та відобразила основну інформацію про нього

Таблиця 4.4 – Тест 1.4. Перевірка вибору двигуна автомобіля

Тест	Перевірка вибору двигуна автомобіля
Модуль	Вибір двигуна
Номер тесту	1.4
Початковий стан системи	Користувач знаходиться на сторінці автомобіля
Вхідні дані	Вибір двигуна зі списку
Опис проведення тесту	Користувач обирає двигун зі списку доступних двигунів
Очікуваний результат	Система відобразить список доступних двигунів для обраного автомобіля
Фактичний результат	Система відобразила список доступних двигунів для обраного автомобіля

Таблиця 4.5 – Тест 1.5. Перевірка вибору трансмісії автомобіля

Тест	Перевірка вибору трансмісії автомобіля
Модуль	Вибір трансмісії
Номер тесту	1.5
Початковий стан системи	Користувач знаходиться на сторінці автомобіля
Вхідні дані	Вибір трансмісії зі списку
Опис проведення тесту	Користувач обирає трансмісію зі списку доступних трансмісій
Очікуваний результат	Система відобразить список доступних трансмісій для обраного автомобіля
Фактичний результат	Система відобразила список доступних трансмісій для обраного автомобіля
Модуль	Вибір трансмісії

Таблиця 4.6 – Тест 1.6. Перевірка підбору розхідників

Тест	Перевірка підбору розхідників
Модуль	Підбір розхідників
Номер тесту	1.6
Початковий стан системи	Користувач обрав двигун та трансмісію для автомобіля
Вхідні дані	Немає
Опис проведення тесту	Користувач натискає кнопку підбору розхідників
Очікуваний результат	Система переведе користувача на сторінку з доступними категоріями запчастин та відобразить їх
Фактичний результат	Система перевела користувача на сторінку з доступними категоріями запчастин та відобразила їх

Таблиця 4.7 – Тест 1.7. Перевірка вибору категорії запчастин

Тест	Перевірка вибору категорії запчастин
Модуль	Вибір категорії запчастин
Номер тесту	1.7
Початковий стан системи	Користувач знаходиться на сторінці з доступними категоріями запчастин
Вхідні дані	Категорія "Масляні фільтри"
Опис проведення тесту	Користувач обирає категорію "Масляні фільтри"
Очікуваний результат	Система переведе користувача на сторінку з доступними запчастинами в даній категорії та відобразить їх опис
Фактичний результат	Система перевела користувача на сторінку з доступними запчастинами в даній категорії та відобразила їх опис

Таблиця 4.8 – Тест 1.8. Перевірка переходу на сторінку з інформацією про компанію

Тест	Перевірка переходу на сторінку з інформацією про компанію
Модуль	Інформація про компанію
Номер тесту	1.8
Початковий стан системи	Користувач знаходиться на будь-якій сторінці веб-застосунку
Вхідні дані	Немає
Опис проведення тесту	Користувач натискає на вкладку "Інформація" в меню зверху
Очікуваний результат	Система переведе користувача на сторінку з інформацією про компанію

Продовження таблиці 4.8

Фактичний результат	Система перевела користувача на сторінку з інформацією про компанію

4.3 Опис контрольного прикладу

Для успішної фіналізації розробки, необхідно протестувати контрольний приклад ПЗ. Далі наведено опис кроків, а на рисунках 4.2 – 4.10 ілюстрації до них.

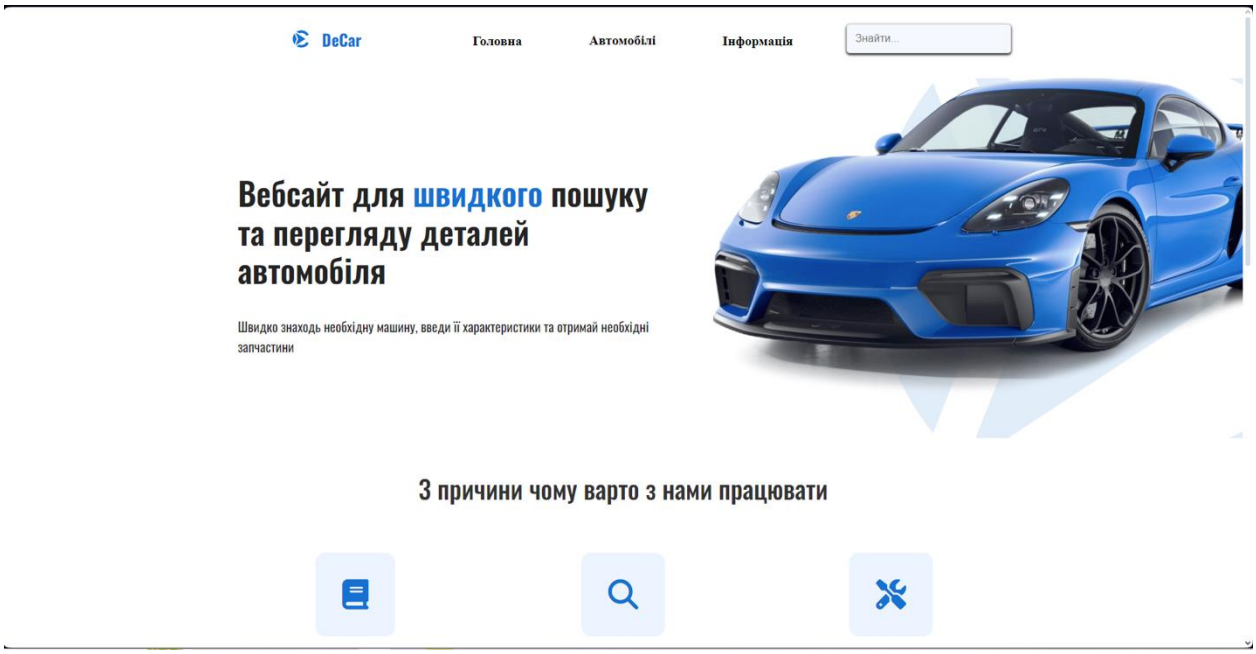

Рисунок 4.2 – Головна сторінка сайту

Для початку спробуємо пошукати автомобіль за моделлю. Для цього натискаємо на поле пошуку та вводимо назву моделі Octavia. Як можна побачити, на рисунку 4.3 система успішно провела пошук моделі в базі даних та запропонувала користувачу варіант.

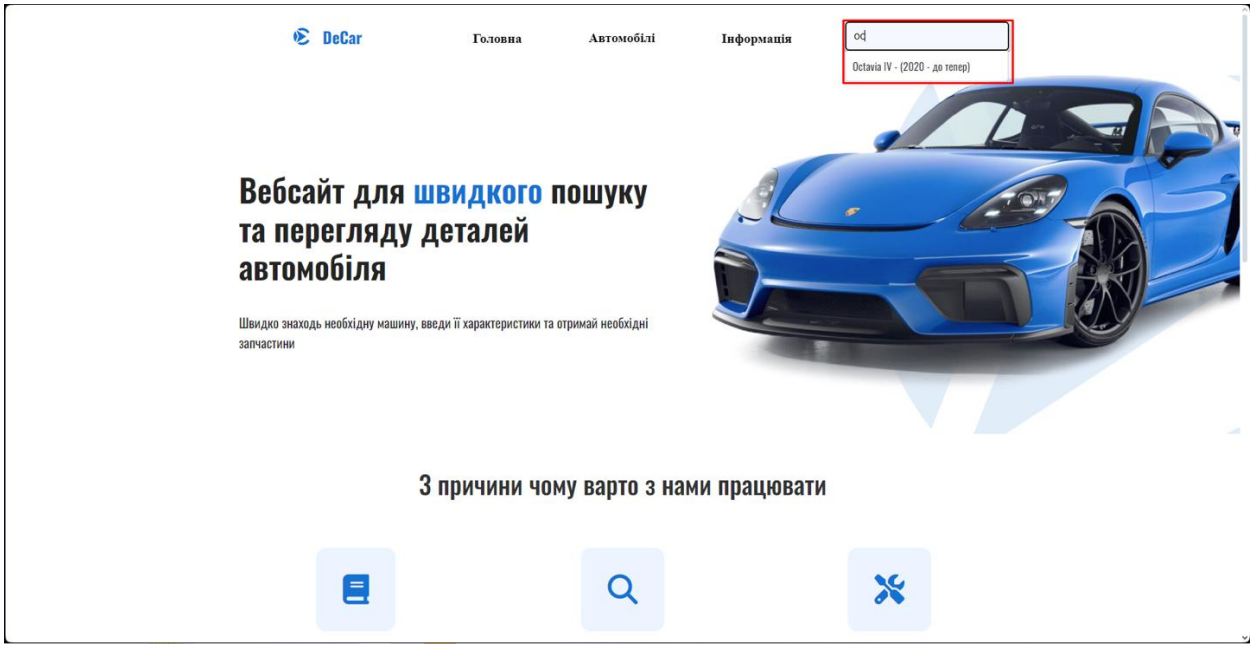

Рисунок 4.3 – Зовнішній вигляд сайту при спробі пошуку

Перейдемо до пошуку за маркою авто. Для цього переходимо на вкладку Автомобілі, та обираємо Volkswagen. Як видно на рисунку 4.4, система успішно відобразила всі доступні моделі авто марки Volkswagen.

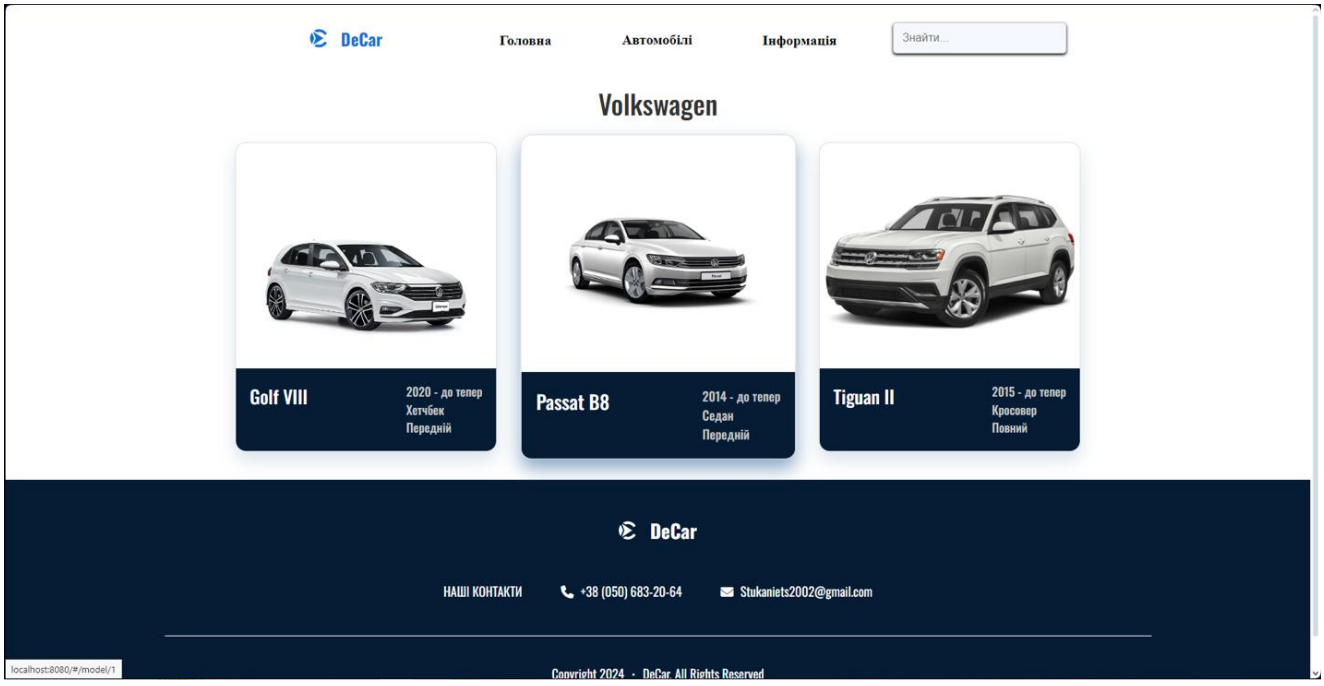

Рисунок 4.4 – зовнішній вигляд сайту при пошуку авто за маркою

Далі обираємо модель яка нас цікавить, а саме, Golf. Як видно на рисунку 4.5, система успішно перевела користувача на сторінку автомобіля відобразила основну інформацію про нього..

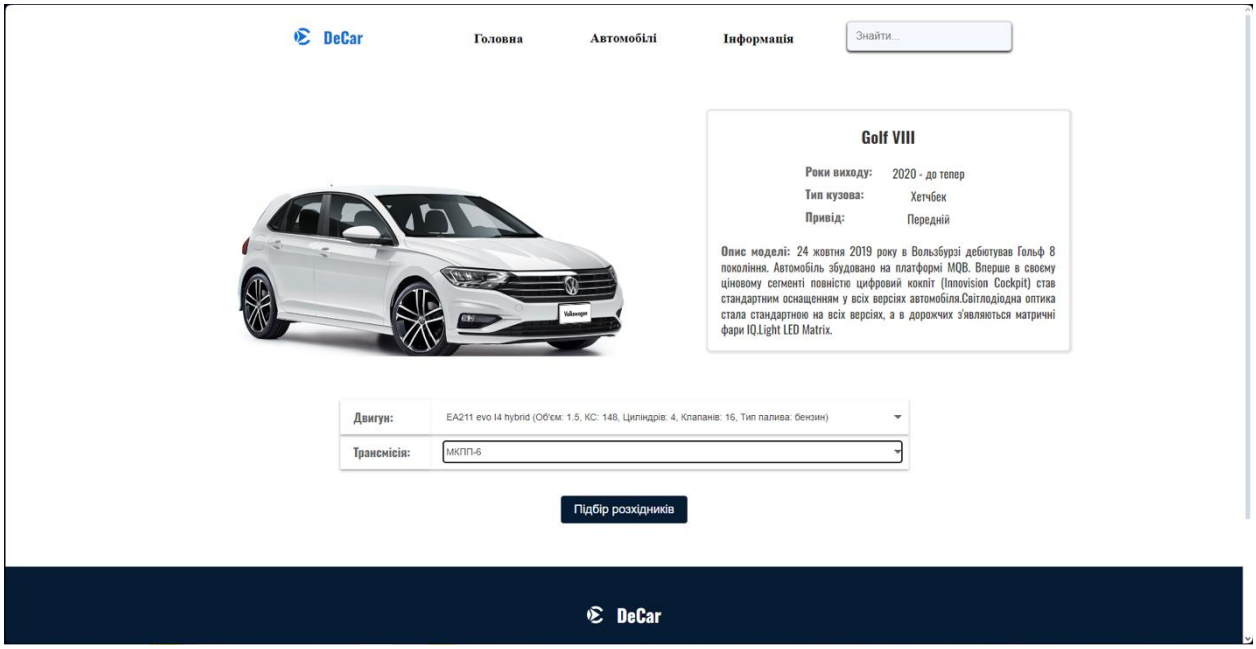

Рисунок 4.5 – Зовнішній вигляд сайту при переході на сторінкку автомобіля

Далі протестуємо поведінку сайту при виборі двигуна авто. Для цього на стрінці з інформацією про автомобіль на випадаючий список під назвою двигун та, як видно на рисунку 4.6, бачимо перед собою список доступних двигунів для даного автомобіля.

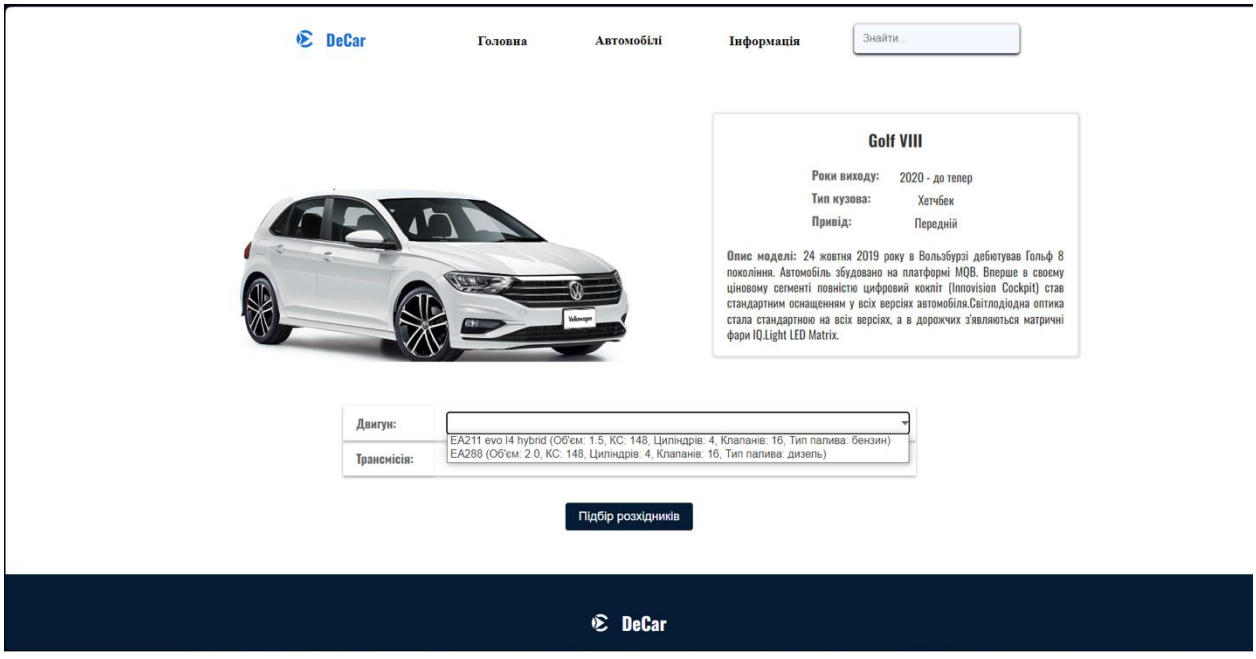

Рисунок 4.6 - Зовнішній вигляд сайту при виборі двигуна

Далі протестуємо поведінку сайту при виборі трансмісії авто. Для цього на сторінці з інформацією про автомобіль на випадаючий список під назвою трансмісія та, як видно на рисунку 4.7, бачимо перед собою список доступних двигунів для даного автомобіля.

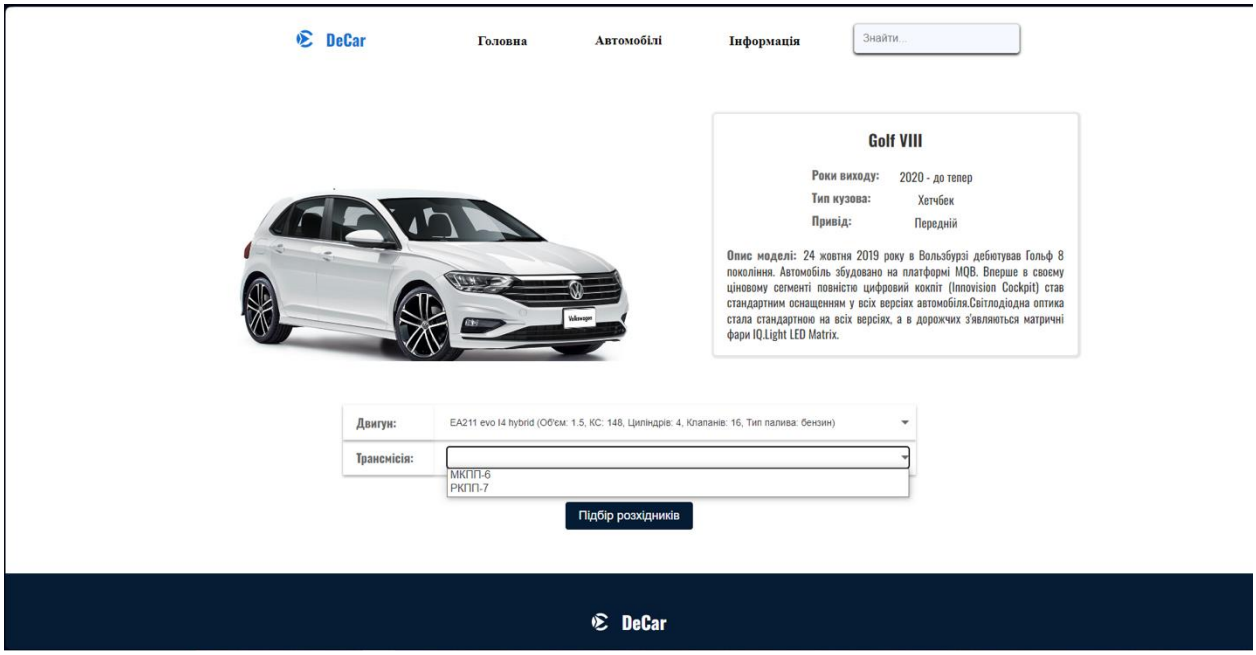

Рисунок 4.7 - Зовнішній вигляд сайту при виборі трансмісії

Після того як було обрано двигун та траснмісію, натискаємо на кнопку підбору розхідників. Як видно на рисунку 4.8, застосунок успішно перевів користувача на сторінку з достуними категоріями запчастин та відобразив їх.

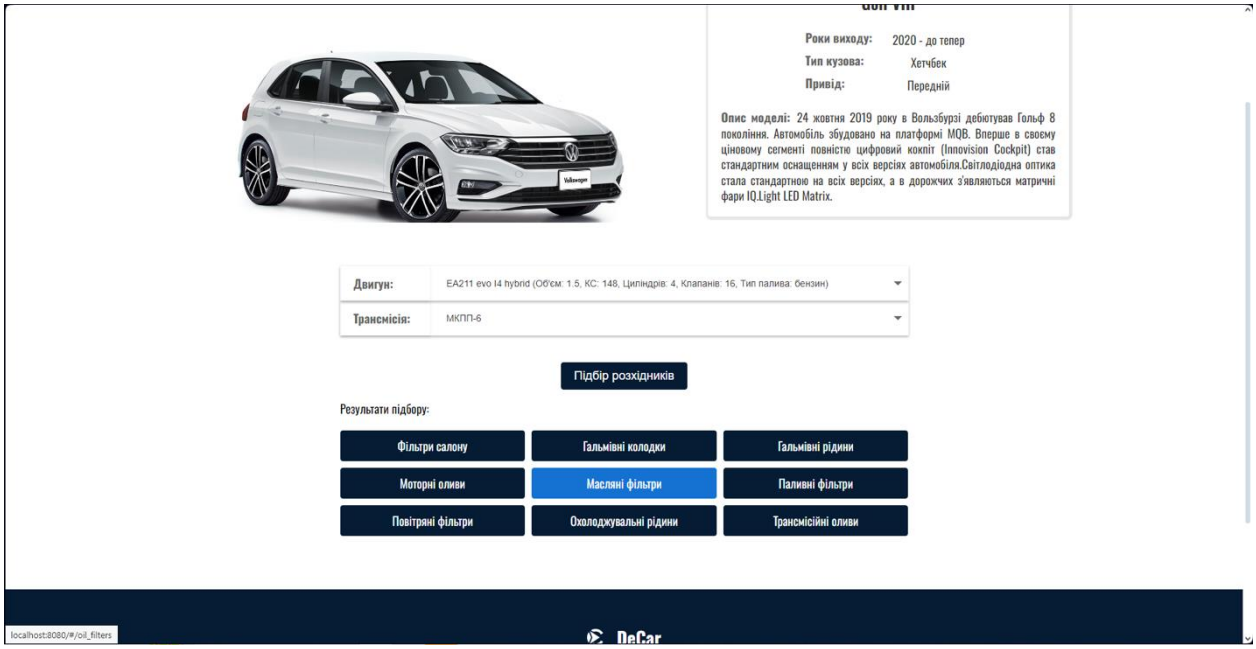

Рисунок 4.8 – Зовнішній вигляд сайту після підбору розхідників

Далі обираємо категорію «Масляні фільтри» та як бачимо на рисунку 4.9, застосунок успішно переводить користувача на сторінку з доступними запчастинами в даній категорії та їх описом.

Рисунок 4.9 – Зовнішній вигляд сторінки з доступними розхідниками

Далі перейдемо на вкладку з інформацією про компанію. Для цього в меню зверху обираємо вкладку «Інформація». Як видно на рисунку 4.10, застосунок успішно перевів нас на сторінку з інформацією про компанію.

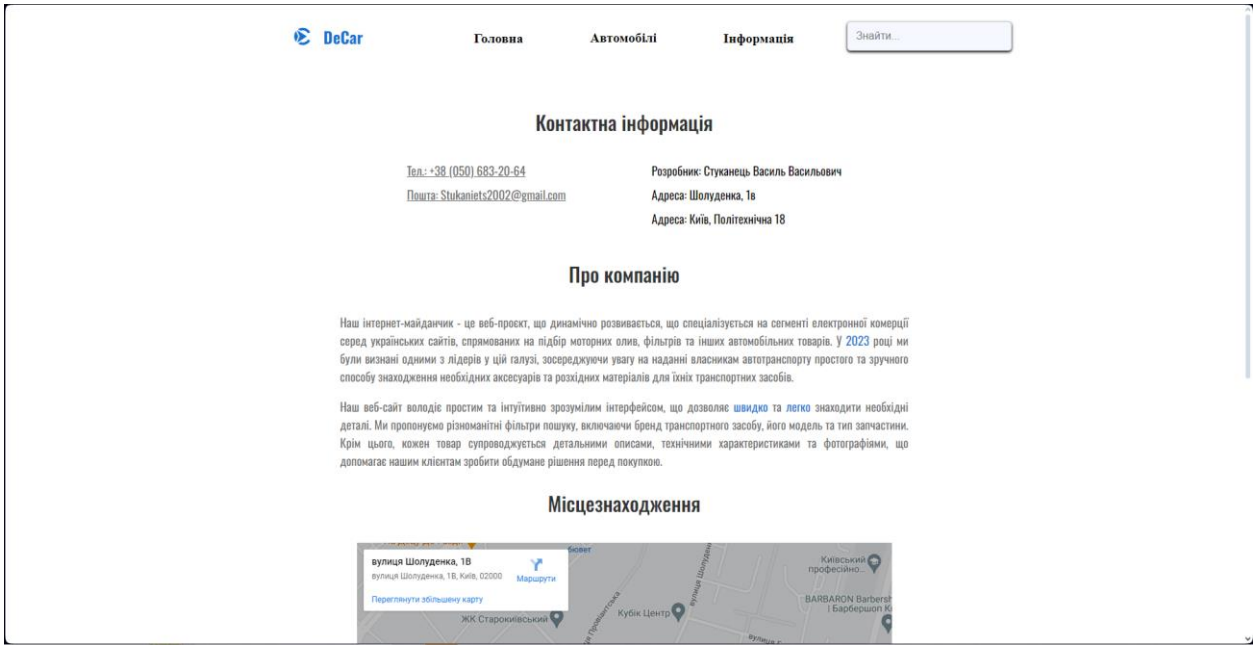

Рисунок 4.10 - Зовнішній вигляд сторінки з інформацією про компанію

Висновки до розділу

У цьому розділі було проведено аналіз якості програмного забезпечення за допомогою спеціалізованих інструментів, а також описано процеси тестування функціональних частин системи. Результати тестування підтвердили відповідність програмного продукту розробленим вимогам та забезпечили високу якість системи. Було описано контрольний приклад, що дозволило оцінити практичну реалізацію функціональних можливостей та забезпечити коректну роботу системи в різних сценаріях використання.

5 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Розгортання програмного забезпечення є ключовим етапом у процесі розробки, оскільки воно дозволяє користувачам взаємодіяти з вашим продуктом у реальному середовищі. У цьому розділі описується процес розгортання нашого додатку, зокрема використання Azure Blob Storage для статичних файлів та Azure App Services для розгортання контейнеризованого бекенду.

Фронтенд частина застосунку складається зі статичних файлів, таких як HTML, CSS та JavaScript, які обслуговуються через Azure Blob Storage. Azure Blob Storage забезпечує зручний та надійний спосіб зберігання та обслуговування статичних вебсайтів. Нижче наведено основні кроки для розгортання фронтенду.

Підготовка файлів:

– Збір усіх необхідних статичних файлів у відповідні каталоги.

– Перевірка файлів на наявність помилок та забезпечення їх коректної роботи.

Налаштування Azure Blob Storage:

– Створення нового контейнера в обліковому записі Azure Storage.

– Завантаження статичних файлів у створений контейнер.

– Налаштування доступу до контейнера, щоб дозволити публічний доступ до файлів.

– Увімкнення функції статичного вебсайту в Azure Blob Storage.

– Вказання основного файлу (наприклад, index.html) та файлу помилки (наприклад, 404.html).

– Публікація URL-адреси сайту, наданої Azure Blob Storage, для доступу користувачів.

Бекенд частину застосунку розгорнуто за допомогою Azure App Services як контейнер. Це забезпечує масштабованість та зручність керування додатком. Основні кроки для розгортання бекенду включають:

– Написання Dockerfile для визначення середовища виконання додатку.

– Побудова Docker-образу та завантаження його в Azure Container Registry (ACR).

– Створення нового екземпляра Azure App Service.

– Конфігурація сервісу для використання Docker-контейнера з ACR.

– Вказання зображення контейнера та тегів для Azure App Service.

– Налаштування середовищних змінних та налаштувань програми через портал Azure.

– Проведення тестування для перевірки коректності роботи бекенду після розгортання.

– Налаштування моніторингу та логування для відстеження продуктивності та виявлення можливих проблем.

Розгортання програмного забезпечення у середовищі Azure забезпечує надійну та масштабовану інфраструктуру для нашого додатку. Використання Azure Blob Storage для статичних файлів та Azure App Services для контейнеризованого бекенду дозволяє швидко та ефективно доставляти оновлення користувачам, забезпечуючи високу продуктивність та стабільність роботи додатку.

5.2 Вимоги до системи користувача

Для забезпечення нормальної роботи користувача із сайтом пошуку автозапчастин, необхідно дотримуватися наступних вимог:

– Процесор: двоядерний процесор з частотою не менше 1.5 ГГц.

– Оперативна пам'ять: Мінімум 4 ГБ.

– Місце на диску: мінімум 500 МБ вільного місця на жорсткому диску.

– Монітор: роздільна здатність екрану не менше 1280x720 пікселів.

– Операційна система: Windows 7 або новіше, Mac OS X 10.10 або новіше, або сучасна дистрибуція Linux.

– Веб-браузер: остання версія Google Chrome, Mozilla Firefox, Microsoft Edge або Safari.

– Додаткове програмне забезпечення: підтримка JavaScript та файлів cookie у браузері.

– Інтернет-з'єднання: швидкість підключення до Інтернету не менше 5 Мбіт/с для стабільної роботи з веб-додатком.

– Навички роботи з комп'ютером: базові навички користування веб-браузером та інтернетом.

Ці вимоги забезпечують оптимальну роботу сайту та комфортне користування для користувачів, дозволяючи використовувати всі функції та можливості, які надає сайт пошуку автозапчастин.

ВИСНОВКИ

В даному дипломному проєкті було проведено аналіз предметної області. Також було розроблено функціональні та нефункціональні вимоги, наведено варіанти використання застосунку та проведено аналіз переваг і недоліків аналогів, що дало змогу приступити до етапу моделювання та розробки програмного забезпечення.

На етапі програмного моделювання та аналізу бізнес-процеси були описані за допомогою моделей BPMN. Ці моделі забезпечили чітке розуміння послідовності дій і взаємодії користувача в застосунку. Під час моделювання було вирішено використовувати архітектуру MVC. Архітектурний патерн MVC дозволив чітко розділити відповідальність між компонентами моделі, представлення та контролера. Цей поділ полегшив організацію коду, зручність обслуговування та масштабованість програми.

На етапі аналізу та тестування якості програмного забезпечення було підкреслено важливість оцінки якості програмного забезпечення та опис проведення процесів оцінки якості коду та мануального тестування. Процеси тестування та контрольний приклад були описані для забезпечення функціональності, надійності та зручності використання застосунку користувачем.

Також було розглянуто процес впровадження та підтримки програмного забезпечення, зосереджуючись на розгортанні програмного забезпечення та постійній підтримці для забезпечення безперебійної роботи програми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) ETKA [Електронний ресурс] // VAG Europa – Режим доступу до ресурсу: https://www.etka.com/etkaportal/en/.
2) Motorad [Електронний ресурс] // Motorad – Режим доступу до ресурсу: https://eu.motorad.com.
3) JavaScript [Електронний ресурс] // Mozilla.– Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
4) Vue [Електронний ресурс] // Vue mastery.– Режим доступу до ресурсу: https://vuejs.org/guide/introduction.
5) Python documentation [Електронний ресурс] // Python Software Foundation.– Режим доступу до ресурсу: https://www.python.org/doc/.
6) Django documentation [Електронний ресурс] // Django Software Foundation. – Режим доступу до ресурсу: https://docs.djangoproject.com/en/5.0/.
7) Django REST [Електронний ресурс] // Encode OSS Ltd. – Режим доступу до ресурсу: https://www.django-rest-framework.org/#development.

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016322934

Дата перевірки:
05.06.2024 17:14:22 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
05.06.2024 17:16:37 EEST

ID користувача:
76913

Назва документа: **ІТ-04_Стуканець_ПЗ**

Кількість сторінок: **54** Кількість слів: **7599** Кількість символів: **61425** Розмір файлу: **5.00 MB** ID файлу: **1016121457**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

12.5%
Схожість

Найбільша схожість: **3.46%** з джерелом з Бібліотеки (ID файлу: **1015069748**)

3.51% Джерела з Інтернету	272	Сторінка 56
12.3% Джерела з Бібліотеки	496	Сторінка 58

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	10 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Вебзастосунок для пошуку автозапчастин

Текст програми

КПІ.ІТ04-9426.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:
_____ Павло РАДІОНОВ

Виконавець:
_____ Василь СТУКАНЕЦЬ

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/Stukanets/diploma
Файл admin.py

```
from django.contrib import admin
from django.contrib.auth import admin as auth_admin
from django.contrib.auth.models import Group, User
from .models import CompanyInfo, CarBrand, CarModel, CarEngine,
CarTransmission, CabinFilter, BrakePad, BrakeFluid, \
    EngineOil, OilFilter, FuelFilter, AirFilter, Coolant, TransmissionOil,
CarModelCabinFilter, CarModelBrakePad, \
    CarModelBrakeFluid, CarEngineEngineOil, CarEngineOilFilter,
CarEngineFuelFilter, CarEngineAirFilter, \
    CarEngineCoolant, CarTransmissionTransmissionOil

admin.site.unregister(Group)


class UserAdmin(auth_admin.UserAdmin):
    model = User

    change_password_form = auth_admin.AdminPasswordChangeForm

    list_display = ['username', 'email', 'is_active', 'is_staff', 'is_superuser']
    search_fields = ['username', 'email']
    readonly_fields = ('last_login', 'date_joined',)
    list_filter = ['is_active', 'is_staff', 'is_superuser']

    ordering = ['-is_staff']

    fieldsets = (('Personal info', {'fields': ('username', 'password', 'email')}),
                 ('Permissions', {'fields': ('is_active', 'is_staff', 'is_superuser')}),
                 ('Important dates', {'fields': ('date_joined', 'last_login')}))


admin.site.unregister(User)
admin.site.register(User, UserAdmin)

admin.site.register(CompanyInfo)
admin.site.register(CarBrand)
admin.site.register(CarModel)
admin.site.register(CarEngine)
admin.site.register(CarTransmission)
admin.site.register(CabinFilter)
admin.site.register(BrakePad)
```

```
admin.site.register(BrakeFluid)
admin.site.register(EngineOil)
admin.site.register(OilFilter)
admin.site.register(FuelFilter)
admin.site.register(AirFilter)
admin.site.register(Coolant)
admin.site.register(TransmissionOil)
admin.site.register(CarModelCabinFilter)
admin.site.register(CarModelBrakePad)
admin.site.register(CarModelBrakeFluid)
admin.site.register(CarEngineEngineOil)
admin.site.register(CarEngineOilFilter)
admin.site.register(CarEngineFuelFilter)
admin.site.register(CarEngineAirFilter)
admin.site.register(CarEngineCoolant)
admin.site.register(CarTransmissionTransmissionOil)
```

Файл apps.py

```
from django.apps import AppConfig



class ApplicationConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'application'
```

Файл models.py

```
from django.db import models
from phonenumber_field.modelfields import PhoneNumberField



class CompanyInfo(models.Model):
    company_name = models.CharField(max_length=50)
    company_owner = models.CharField(max_length=100)
    company_email = models.EmailField(max_length=100)
    company_phone_number = PhoneNumberField(null=False, blank=False,
unique=False)
    company_first_address = models.CharField(max_length=100)
```

```
    company_second_address = models.CharField(max_length=100)
    company_address_google_map = models.TextField()

    class Meta:
        db_table = 'company_info'
        ordering = ['company_name']
        managed = True
        verbose_name = 'Company information'
        verbose_name_plural = 'Company information'

    def __str__(self):
        return f'{str(self.company_name)}'


class CarBrand(models.Model):
    company = models.ForeignKey(CompanyInfo, on_delete=models.CASCADE,
related_name='car_brands_info')
    brand_name = models.CharField(max_length=50)
    brand_location = models.CharField(max_length=50)
    brand_year_of_establishment = models.DateField()
    brand_description = models.TextField()
    brand_logo_image = models.ImageField(null=True, blank=True,
upload_to='images/')
    brand_website_link = models.TextField()

    class Meta:
        db_table = 'car_brands'
        ordering = ['brand_name']
        managed = True
        verbose_name = 'Car brands'
```

```
        verbose_name_plural = 'Car brands'

    def __str__(self):
        return f'{str(self.brand_name)}'


class CarModel(models.Model):
    car_brand = models.ForeignKey(CarBrand, on_delete=models.CASCADE,
related_name='car_models_info')
    model_name = models.CharField(max_length=50)
    model_years_of_production = models.CharField(max_length=20)
    model_body_type = models.CharField(max_length=50)
    model_wheel_drive_type = models.CharField(max_length=50)
    model_description = models.TextField()
    model_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'car_models'
        ordering = ['model_name']
        managed = True
        verbose_name = 'Car models'
        verbose_name_plural = 'Car models'

    def __str__(self):
        return f'{str(self.model_name)}'


class CarEngine(models.Model):
```

```
    car_model = models.ForeignKey(CarModel, on_delete=models.CASCADE,
related_name='car_engines_info')
    engine_name = models.CharField(max_length=50)
    engine_capacity = models.DecimalField(max_digits=2, decimal_places=1)
    engine_hp = models.IntegerField()
    engine_number_of_cylinders = models.IntegerField()
    engine_number_of_valves = models.IntegerField()
    engine_fuel_type = models.CharField(max_length=50)

    class Meta:
        db_table = 'car_engines'
        ordering = ['engine_name']
        managed = True
        verbose_name = 'Car engines'
        verbose_name_plural = 'Car engines'

    def __str__(self):
        return f'{str(self.engine_name)}'


class CarTransmission(models.Model):
    car_model = models.ForeignKey(CarModel, on_delete=models.CASCADE,
related_name='car_transmissions_info')
    transmission_type = models.CharField(max_length=20)

    class Meta:
        db_table = 'car_transmissions'
        ordering = ['transmission_type']
        managed = True
        verbose_name = 'Car transmissions'
```

```
        verbose_name_plural = 'Car transmissions'

    def __str__(self):
        return f'{str(self.transmission_type)}'


class CabinFilter(models.Model):
    cabin_filter_name = models.CharField(max_length=50)
    cabin_filter_type = models.CharField(max_length=50)
    cabin_filter_brand = models.CharField(max_length=50)
    cabin_filter_description = models.TextField()
    cabin_filter_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'cabin_filters'
        ordering = ['cabin_filter_name']
        managed = True
        verbose_name = 'Cabin filters'
        verbose_name_plural = 'Cabin filters'

    def __str__(self):
        return f'{str(self.cabin_filter_name)}'


class BrakePad(models.Model):
    brake_pad_name = models.CharField(max_length=50)
    brake_pad_type = models.CharField(max_length=50)
    brake_pad_brand = models.CharField(max_length=50)
    brake_pad_description = models.TextField()
```

```
    brake_pad_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'brake_pads'
        ordering = ['brake_pad_name']
        managed = True
        verbose_name = 'Brake pads'
        verbose_name_plural = 'Brake pads'

    def __str__(self):
        return f'{str(self.brake_pad_name)}'


class BrakeFluid(models.Model):
    brake_fluid_name = models.CharField(max_length=50)
    brake_fluid_type = models.CharField(max_length=50)
    brake_fluid_brand = models.CharField(max_length=50)
    brake_fluid_description = models.TextField()
    brake_fluid_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'brake_fluids'
        ordering = ['brake_fluid_name']
        managed = True
        verbose_name = 'Brake fluids'
        verbose_name_plural = 'Brake fluids'

    def __str__(self):
```

```
        return f'{str(self.brake_fluid_name)}'


class EngineOil(models.Model):
    engine_oil_name = models.CharField(max_length=50)
    engine_oil_type = models.CharField(max_length=50)
    engine_oil_density = models.CharField(max_length=50)
    engine_oil_tolerances = models.CharField(max_length=50)
    engine_oil_brand = models.CharField(max_length=50)
    engine_oil_description = models.TextField()
    engine_oil_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'engine_oils'
        ordering = ['engine_oil_name']
        managed = True
        verbose_name = 'Engine oils'
        verbose_name_plural = 'Engine oils'

    def __str__(self):
        return f'{str(self.engine_oil_name)}'


class OilFilter(models.Model):
    oil_filter_name = models.CharField(max_length=50)
    oil_filter_brand = models.CharField(max_length=50)
    oil_filter_description = models.TextField()
    oil_filter_image = models.ImageField(null=True, blank=True,
upload_to='images/')
```

```
    class Meta:
        db_table = 'oil_filters'
        ordering = ['oil_filter_name']
        managed = True
        verbose_name = 'Oil filters'
        verbose_name_plural = 'Oil filters'

    def __str__(self):
        return f'{str(self.oil_filter_name)}'


class FuelFilter(models.Model):
    fuel_filter_name = models.CharField(max_length=50)
    fuel_filter_type = models.CharField(max_length=50)
    fuel_filter_brand = models.CharField(max_length=50)
    fuel_filter_description = models.TextField()
    fuel_filter_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'fuel_filters'
        ordering = ['fuel_filter_name']
        managed = True
        verbose_name = 'Fuel filters'
        verbose_name_plural = 'Fuel filters'

    def __str__(self):
        return f'{str(self.fuel_filter_name)}'
```

```
class AirFilter(models.Model):
    air_filter_name = models.CharField(max_length=50)
    air_filter_type = models.CharField(max_length=50)
    air_filter_brand = models.CharField(max_length=50)
    air_filter_description = models.TextField()
    air_filter_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'air_filters'
        ordering = ['air_filter_name']
        managed = True
        verbose_name = 'Air filters'
        verbose_name_plural = 'Air filters'

    def __str__(self):
        return f'{str(self.air_filter_name)}'


class Coolant(models.Model):
    coolant_name = models.CharField(max_length=50)
    coolant_type = models.CharField(max_length=50)
    coolant_color = models.CharField(max_length=50)
    coolant_brand = models.CharField(max_length=50)
    coolant_description = models.TextField()
    coolant_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
```

```
        db_table = 'coolants'
        ordering = ['coolant_name']
        managed = True
        verbose_name = 'Coolants'
        verbose_name_plural = 'Coolants'

    def __str__(self):
        return f'{str(self.coolant_name)}'


class TransmissionOil(models.Model):
    transmission_oil_name = models.CharField(max_length=50)
    transmission_oil_type = models.CharField(max_length=50)
    transmission_oil_atf_type = models.CharField(max_length=50)
    transmission_oil_brand = models.CharField(max_length=50)
    transmission_oil_description = models.TextField()
    transmission_oil_image = models.ImageField(null=True, blank=True,
upload_to='images/')

    class Meta:
        db_table = 'transmission_oils'
        ordering = ['transmission_oil_name']
        managed = True
        verbose_name = 'Transmission oils'
        verbose_name_plural = 'Transmission oils'

    def __str__(self):
        return f'{str(self.transmission_oil_name)}'
```

```
class CarModelCabinFilter(models.Model):
    car_model = models.ForeignKey(CarModel, on_delete=models.CASCADE)
    cabin_filter = models.ForeignKey(CabinFilter, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_model_cabin_filters'
        ordering = ['car_model']
        managed = True
        verbose_name = 'Car model - cabin filters'
        verbose_name_plural = 'Car model - cabin filters'

    def __str__(self):
        return f'{str(self.car_model)} - {str(self.cabin_filter)}'


class CarModelBrakePad(models.Model):
    car_model = models.ForeignKey(CarModel, on_delete=models.CASCADE)
    brake_pad = models.ForeignKey(BrakePad, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_model_brake_pads'
        ordering = ['car_model']
        managed = True
        verbose_name = 'Car model - brake pads'
        verbose_name_plural = 'Car model - brake pads'

    def __str__(self):
        return f'{str(self.car_model)} - {str(self.brake_pad)}'
```

```
class CarModelBrakeFluid(models.Model):
    car_model = models.ForeignKey(CarModel, on_delete=models.CASCADE)
    brake_fluid = models.ForeignKey(BrakeFluid, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_model_brake_fluids'
        ordering = ['car_model']
        managed = True
        verbose_name = 'Car model - brake fluids'
        verbose_name_plural = 'Car model - brake fluids'

    def __str__(self):
        return f'{str(self.car_model)} - {str(self.brake_fluid)}'


class CarEngineEngineOil(models.Model):
    car_engine = models.ForeignKey(CarEngine, on_delete=models.CASCADE)
    engine_oil = models.ForeignKey(EngineOil, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_engine_engine_oils'
        ordering = ['car_engine']
        managed = True
        verbose_name = 'Car engine - engine oils'
        verbose_name_plural = 'Car engine - engine oils'

    def __str__(self):
        return f'{str(self.car_engine)} - {str(self.engine_oil)}'
```

```
class CarEngineOilFilter(models.Model):
    car_engine = models.ForeignKey(CarEngine, on_delete=models.CASCADE)
    oil_filter = models.ForeignKey(OilFilter, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_engine_oil_filters'
        ordering = ['car_engine']
        managed = True
        verbose_name = 'Car engine - oil filters'
        verbose_name_plural = 'Car engine - oil filters'

    def __str__(self):
        return f'{str(self.car_engine)} - {str(self.oil_filter)}'


class CarEngineFuelFilter(models.Model):
    car_engine = models.ForeignKey(CarEngine, on_delete=models.CASCADE)
    fuel_filter = models.ForeignKey(FuelFilter, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_engine_fuel_filters'
        ordering = ['car_engine']
        managed = True
        verbose_name = 'Car engine - fuel filters'
        verbose_name_plural = 'Car engine - fuel filters'

    def __str__(self):
        return f'{str(self.car_engine)} - {str(self.fuel_filter)}'
```

```
class CarEngineAirFilter(models.Model):
    car_engine = models.ForeignKey(CarEngine, on_delete=models.CASCADE)
    air_filter = models.ForeignKey(AirFilter, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_engine_air_filters'
        ordering = ['car_engine']
        managed = True
        verbose_name = 'Car engine - air filters'
        verbose_name_plural = 'Car engine - air filters'

    def __str__(self):
        return f'{str(self.car_engine)} - {str(self.air_filter)}'


class CarEngineCoolant(models.Model):
    car_engine = models.ForeignKey(CarEngine, on_delete=models.CASCADE)
    coolant = models.ForeignKey(Coolant, on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_engine_coolants'
        ordering = ['car_engine']
        managed = True
        verbose_name = 'Car engine - coolants'
        verbose_name_plural = 'Car engine - coolants'

    def __str__(self):
        return f'{str(self.car_engine)} - {str(self.coolant)}'
```

```
class CarTransmissionTransmissionOil(models.Model):
    car_transmission = models.ForeignKey(CarTransmission,
on_delete=models.CASCADE)
    transmission_oil = models.ForeignKey(TransmissionOil,
on_delete=models.CASCADE)

    class Meta:
        db_table = 'car_transmission_transmission_oils'
        ordering = ['car_transmission']
        managed = True
        verbose_name = 'Car transmission - transmission oils'
        verbose_name_plural = 'Car transmission - transmission oils'

    def __str__(self):
        return f'{str(self.car_transmission)} - {str(self.transmission_oil)}'
```

Файл serializer.py

```
from rest_framework.serializers import ModelSerializer
from .models import (CompanyInfo, CarBrand, CarModel, CarEngine,
CarTransmission, CabinFilter, BrakePad, BrakeFluid,
                     EngineOil, OilFilter, FuelFilter, AirFilter, Coolant, TransmissionOil)


class CompanyInfoSerializer(ModelSerializer):
    class Meta:
        model = CompanyInfo
        fields = ['company_name',
                  'company_email',
                  'company_phone_number']
```

```
class CarBrandsNameInfoSerializer(ModelSerializer):
    class Meta:
        model = CarBrand
        fields = ['id',
                  'brand_name']


class CarModelsInfoSerializer(ModelSerializer):
    class Meta:
        model = CarModel
        fields = ['id',
                  'model_name',
                  'model_years_of_production',
                  'model_body_type',
                  'model_wheel_drive_type',
                  'model_description',
                  'model_image']


class CompanyExtendedInfoSerializer(ModelSerializer):
    class Meta:
        model = CompanyInfo
        fields = ['company_name',
                  'company_owner',
                  'company_email',
                  'company_phone_number',
                  'company_first_address',
                  'company_second_address',
                  'company_address_google_map']
```

```
class CarBrandsInfoSerializer(ModelSerializer):
    car_models_info = CarModelsInfoSerializer(many=True, read_only=True)

    class Meta:
        model = CarBrand
        fields = ['id',
                  'brand_name',
                  'brand_location',
                  'brand_year_of_establishment',
                  'brand_description',
                  'brand_logo_image',
                  'brand_website_link',
                  'car_models_info']


class CarEnginesInfoSerializer(ModelSerializer):
    class Meta:
        model = CarEngine
        fields = ['id',
                  'engine_name',
                  'engine_capacity',
                  'engine_hp',
                  'engine_number_of_cylinders',
                  'engine_number_of_valves',
                  'engine_fuel_type']


class CarTransmissionsInfoSerializer(ModelSerializer):
```

```
    class Meta:
        model = CarTransmission
        fields = ['id',
                  'transmission_type']


class CarModelsEngineAndTransmissionInfoSerializer(ModelSerializer):
    car_engines_info = CarEnginesInfoSerializer(many=True, read_only=True)
    car_transmissions_info = CarTransmissionsInfoSerializer(many=True,
read_only=True)

    class Meta:
        model = CarModel
        fields = ['id',
                  'model_name',
                  'model_years_of_production',
                  'model_body_type',
                  'model_wheel_drive_type',
                  'model_description',
                  'model_image',
                  'car_engines_info',
                  'car_transmissions_info']


class CabinFiltersInfoSerializer(ModelSerializer):
    class Meta:
        model = CabinFilter
        fields = ['id',
                  'cabin_filter_name',
                  'cabin_filter_type',
```

```
            'cabin_filter_brand',
            'cabin_filter_description',
            'cabin_filter_image']


class BrakePadsInfoSerializer(ModelSerializer):
    class Meta:
        model = BrakePad
        fields = ['id',
            'brake_pad_name',
            'brake_pad_type',
            'brake_pad_brand',
            'brake_pad_description',
            'brake_pad_image']


class BrakeFluidsInfoSerializer(ModelSerializer):
    class Meta:
        model = BrakeFluid
        fields = ['id',
            'brake_fluid_name',
            'brake_fluid_type',
            'brake_fluid_brand',
            'brake_fluid_description',
            'brake_fluid_image']


class EngineOilsInfoSerializer(ModelSerializer):
    class Meta:
        model = EngineOil
```

```
        fields = ['id',
                'engine_oil_name',
                'engine_oil_type',
                'engine_oil_density',
                'engine_oil_tolerances',
                'engine_oil_brand',
                'engine_oil_description',
                'engine_oil_image']


class OilFiltersInfoSerializer(ModelSerializer):
    class Meta:
        model = OilFilter
        fields = ['id',
                'oil_filter_name',
                'oil_filter_brand',
                'oil_filter_description',
                'oil_filter_image']


class FuelFiltersInfoSerializer(ModelSerializer):
    class Meta:
        model = FuelFilter
        fields = ['id',
                'fuel_filter_name',
                'fuel_filter_type',
                'fuel_filter_brand',
                'fuel_filter_description',
                'fuel_filter_image']
```

```
class AirFiltersInfoSerializer(ModelSerializer):
    class Meta:
        model = AirFilter
        fields = ['id',
                'air_filter_name',
                'air_filter_type',
                'air_filter_brand',
                'air_filter_description',
                'air_filter_image']


class CoolantsInfoSerializer(ModelSerializer):
    class Meta:
        model = Coolant
        fields = ['id',
                'coolant_name',
                'coolant_type',
                'coolant_color',
                'coolant_brand',
                'coolant_description',
                'coolant_image']


class TransmissionOilsInfoSerializer(ModelSerializer):
    class Meta:
        model = TransmissionOil
        fields = ['id',
                'transmission_oil_name',
                'transmission_oil_type',
```

```
            'transmission_oil_atf_type',
            'transmission_oil_brand',
            'transmission_oil_description',
            'transmission_oil_image']
```

Файл urls.py

```
from django.urls import path
from .views import (HomePageView, InformationPageView, BrandPageView,
ModelPageView, SelectedCabinFilters,
                    SelectedBrakePads, SelectedBrakeFluids, SelectedEngineOils,
SelectedOilFilters, SelectedFuelFilters,
                    SelectedAirFilters, SelectedCoolants, SelectedTransmissionOils)


urlpatterns = [
    path('home/', HomePageView.as_view()),
    path('company_information/', InformationPageView.as_view()),
    path('brands/', BrandPageView.as_view()),
    path('models/', ModelPageView.as_view()),
    path('selected_cabin_filters/', SelectedCabinFilters.as_view()),
    path('selected_brake_pads/', SelectedBrakePads.as_view()),
    path('selected_brake_fluids/', SelectedBrakeFluids.as_view()),
    path('selected_engine_oils/', SelectedEngineOils.as_view()),
    path('selected_oil_filters/', SelectedOilFilters.as_view()),
    path('selected_fuel_filters/', SelectedFuelFilters.as_view()),
    path('selected_air_filters/', SelectedAirFilters.as_view()),
    path('selected_coolants/', SelectedCoolants.as_view()),
    path('selected_transmission_oils/', SelectedTransmissionOils.as_view())
]
```

Файл views.py

```
from django.db.models import Q
from rest_framework.views import APIView
from rest_framework.response import Response
from .models import (CompanyInfo, CarBrand, CarModel, CarModelCabinFilter,
CabinFilter, CarModelBrakePad, BrakePad,
                     CarModelBrakeFluid, BrakeFluid, CarEngineEngineOil, EngineOil,
CarEngineOilFilter, OilFilter,
                     CarEngineFuelFilter, FuelFilter, CarEngineAirFilter, AirFilter,
CarEngineCoolant, Coolant,
                     CarTransmissionTransmissionOil, TransmissionOil)
from .serializer import (CompanyInfoSerializer, CarBrandsNameInfoSerializer,
CarModelsInfoSerializer,
                         CompanyExtendedInfoSerializer, CarBrandsInfoSerializer,
                         CarModelsEngineAndTransmissionInfoSerializer,
CabinFiltersInfoSerializer,
                         BrakePadsInfoSerializer, BrakeFluidsInfoSerializer,
EngineOilsInfoSerializer,
                         OilFiltersInfoSerializer, FuelFiltersInfoSerializer,
AirFiltersInfoSerializer,
                         CoolantsInfoSerializer, TransmissionOilsInfoSerializer)


class CompanyInformation:
    @staticmethod
    def get():
        company_info = CompanyInfoSerializer(CompanyInfo.objects, many=True)
        return company_info.data


class CarBrandName:
```

```
    @staticmethod
    def get():
        car_brand_name_ = CarBrandsNameInfoSerializer(CarBrand.objects,
many=True)
        return car_brand_name_.data


class CarModelInformation:
    @staticmethod
    def get():
        car_model_info = CarModelsInfoSerializer(CarModel.objects, many=True)
        return car_model_info.data


class Top3CarsInformation:
    @staticmethod
    def get():
        car_model_info = CarModelsInfoSerializer(CarModel.objects.filter(Q(id=4) |
Q(id=5) | Q(id=10)), many=True)
        return car_model_info.data


class HomePageView(APIView):
    @staticmethod
    def get(request):
        company_info = CompanyInformation.get()
        car_brand_name_info = CarBrandName.get()
        car_model_info = CarModelInformation.get()
        top_3_cars_info = Top3CarsInformation.get()
        return Response({'company_info': company_info,
```

```
                'car_brand_name_info': car_brand_name_info,
                'car_models_info': car_model_info,
                'top_3_cars_info': top_3_cars_info})


class CompanyExtendedInformation:
    @staticmethod
    def get():
        company_extended_info =
CompanyExtendedInfoSerializer(CompanyInfo.objects, many=True)
        return company_extended_info.data


class InformationPageView(APIView):
    @staticmethod
    def get(request):
        company_extended_info = CompanyExtendedInformation.get()
        return Response({'company_extended_info': company_extended_info})


class CarBrandInformation:
    @staticmethod
    def get():
        car_brand_info = CarBrandsInfoSerializer(CarBrand.objects, many=True)
        return car_brand_info.data


class BrandPageView(APIView):
    @staticmethod
    def get(request):
```

```
        car_brand_info = CarBrandInformation.get()
        return Response({'car_brand_info': car_brand_info})


class CarModelsEngineAndTransmissionInformation:
    @staticmethod
    def get():
        car_model_engine_and_transmission_info =
CarModelsEngineAndTransmissionInfoSerializer(
            CarModel.objects, many=True)
        return car_model_engine_and_transmission_info.data


class ModelPageView(APIView):
    @staticmethod
    def get(request):
        car_model_engine_and_transmission_info =
CarModelsEngineAndTransmissionInformation.get()
        return Response({'car_model_engines_and_transmissions_info':
car_model_engine_and_transmission_info})


class SelectedCabinFilters(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        model_id = post_data.get('model_id')

        selected_cabin_filters_id = [elem.cabin_filter.id for elem in
CarModelCabinFilter.objects.filter(
```

```
            car_model_id=model_id)]
        cabin_filter_info =
CabinFiltersInfoSerializer(CabinFilter.objects.filter(id__in=selected_cabin_filters_
id),
                                           many=True).data
        return Response({'cabin_filter_info': cabin_filter_info})


class SelectedBrakePads(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        model_id = post_data.get('model_id')

        selected_brake_pads_id = [elem.brake_pad.id for elem in
CarModelBrakePad.objects.filter(
            car_model_id=model_id)]
        brake_pads_info =
BrakePadsInfoSerializer(BrakePad.objects.filter(id__in=selected_brake_pads_id),
                                          many=True).data
        return Response({'brake_pads_info': brake_pads_info})


class SelectedBrakeFluids(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        model_id = post_data.get('model_id')
```

```
        selected_brake_fluids_id = [elem.brake_fluid.id for elem in
CarModelBrakeFluid.objects.filter(
            car_model_id=model_id)]
        brake_fluids_info =
BrakeFluidsInfoSerializer(BrakeFluid.objects.filter(id__in=selected_brake_fluids_
id),
                                            many=True).data
        return Response({'brake_fluids_info': brake_fluids_info})


class SelectedEngineOils(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        engine_id = post_data.get('engine_id')

        selected_engine_oils_id = [elem.engine_oil.id for elem in
CarEngineEngineOil.objects.filter(
            car_engine_id=engine_id)]
        engine_oils_info =
EngineOilsInfoSerializer(EngineOil.objects.filter(id__in=selected_engine_oils_id),
                                            many=True).data
        return Response({'engine_oils_info': engine_oils_info})


class SelectedOilFilters(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        engine_id = post_data.get('engine_id')
```

```
        selected_oil_filters_id = [elem.oil_filter.id for elem in
CarEngineOilFilter.objects.filter(
            car_engine_id=engine_id)]
        oil_filters_info =
OilFiltersInfoSerializer(OilFilter.objects.filter(id__in=selected_oil_filters_id),
                                                  many=True).data
        return Response({'oil_filters_info': oil_filters_info})


class SelectedFuelFilters(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        engine_id = post_data.get('engine_id')

        selected_fuel_filters_id = [elem.fuel_filter.id for elem in
CarEngineFuelFilter.objects.filter(
            car_engine_id=engine_id)]
        fuel_filters_info =
FuelFiltersInfoSerializer(FuelFilter.objects.filter(id__in=selected_fuel_filters_id),
                                                  many=True).data
        return Response({'fuel_filters_info': fuel_filters_info})


class SelectedAirFilters(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        engine_id = post_data.get('engine_id')
```

```
        selected_air_filters_id = [elem.air_filter.id for elem in
CarEngineAirFilter.objects.filter(
            car_engine_id=engine_id)]
        air_filters_info =
AirFiltersInfoSerializer(AirFilter.objects.filter(id__in=selected_air_filters_id),
                                                  many=True).data
        return Response({'air_filters_info': air_filters_info})


class SelectedCoolants(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        engine_id = post_data.get('engine_id')

        selected_coolants_id = [elem.coolant.id for elem in
CarEngineCoolant.objects.filter(
            car_engine_id=engine_id)]
        coolants_info =
CoolantsInfoSerializer(Coolant.objects.filter(id__in=selected_coolants_id),
                                              many=True).data
        return Response({'coolants_info': coolants_info})


class SelectedTransmissionOils(APIView):
    @staticmethod
    def post(request):
        post_data = request.data
        transmission_id = post_data.get('transmission_id')
```

```
        selected_transmission_oils_id = [elem.transmission_oil.id for elem in
                                         CarTransmissionTransmissionOil.objects.filter(
                                             car_transmission_id=transmission_id)]
        transmission_oils_info =
TransmissionOilsInfoSerializer(TransmissionOil.objects.filter(
            id__in=selected_transmission_oils_id), many=True).data
        return Response({'transmission_oils_info': transmission_oils_info})
```

Файл asgi.py

```
"""
ASGI config for settings project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'settings.settings')

application = get_asgi_application()
```

Файл settings.py

```
"""
Django settings for settings project.

Generated by 'django-admin startproject' using Django 5.0.3.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-g050q$$-6s$&$$c9b9_-
plvisl51%))9ye_#%b3m^ti*0$)%63'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']

CSRF_TRUSTED_ORIGINS = ['https://*', 'http://*', 'https://diploma-vasyl-
api.azurewebsites.net']
CSRF_ALLOWED_ORIGINS = ['https://*', 'http://*', 'https://diploma-vasyl-
api.azurewebsites.net']
```

```
CORS_ORIGINS_WHITELIST = ['https://*', 'http://*', 'https://diploma-vasyl-api.azurewebsites.net']
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'application',
    'rest_framework',
    'phonenumber_field',
    'corsheaders'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
CORS_ORIGIN_ALLOW_ALL = True

ROOT_URLCONF = 'settings.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'settings.wsgi.application'



# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
```

```
        'NAME': 'DeCar_Database',
        'USER': 'vasyl',
        'PASSWORD': 'Password!',
        'HOST': 'diploma-vasyl.mysql.database.azure.com',
        'PORT': '3306'
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Kyiv'

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Файл urls.py

```
"""
URL configuration for settings project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.0/topics/http/urls/
Examples:
```

```
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('application.urls')),
]
```

Файл wsgi.py

```
"""
WSGI config for settings project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'settings.settings')

application = get_wsgi_application()
```

Файл Dockerfile

```
FROM python:3.11-bullseye
WORKDIR /app
COPY . .
RUN pip3 install -r requirements.txt
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

Файл manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'settings.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':
    main()
```

Файл requiremets.txt

```
asgiref==3.8.1
colorama==0.4.6
Django==5.0.3
django-cors-headers==4.3.1
django-phonenumber-field==7.3.0
django-phonenumbers==1.0.1
django-stubs==4.2.7
django-stubs-ext==4.2.7
djangorestframework==3.15.1
mysql-connector-python==8.3.0
mysqlclient==2.2.4
phonenumbers==8.13.33
pillow==10.2.0
sqlparse==0.4.4
types-pytz==2024.1.0.20240203
types-PyYAML==6.0.12.20240311
typing_extensions==4.10.0
tzdata==2024.1
```

Файл test.py

```
import sys
print(sys.executable)
```

Файл App.vue

```
<template>
  <fullHeader />
  <router-view></router-view>
  <fullFooter />
```

```
</template>

<script>
import fullHeader from './components/fullHeader.vue'
import fullFooter from './components/fullFooter.vue'

export default {
  components: {
    fullHeader,
    fullFooter
  }
}
</script>

<style>
body {
  margin: 0 auto;
}

#app {
  font-family: 'Oswald', sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  margin-top: 10px;
}
</style>
```

Файл main.js

```
import { createApp } from 'vue'
import router from './router'
import App from './App.vue'
```

```
const app = createApp(App);
app.use(router);
app.mount('#app');
```

Файл router.js

```
import { createRouter, createWebHashHistory } from 'vue-router';
import mainPC from './components/mainPage.vue';
import model from './components/modelPage.vue';
import brands from './components/brandsPage.vue';
import companyInformation from './components/companyInformation.vue';
import CabinFiltersPage from
'./components/part_components/Page_cabin_filters.vue';
import brakePads from './components/part_components/Page_brake_pads.vue';
import brakeFluids from './components/part_components/Page_brake_fluids.vue';
import engineOils from './components/part_components/Page_engine_oils.vue';
import oilFilters from './components/part_components/Page_oil_filters.vue';
import fuelFilters from './components/part_components/Page_fuel_filters.vue';
import airFilters from './components/part_components/Page_air_filters.vue';
import coolants from './components/part_components/Page_coolants.vue';
import transmissionOils from
'./components/part_components/Page_transmission_oils.vue';

const router = createRouter({
    history: createWebHashHistory(),
    routes: [
        { path: '/', component: mainPC },
        { path: '/brands/:id', component: brands, props: true },
        { path: '/model/:id', name: 'model', component: model, props: true },
        { path: '/cabin_filters', name: 'CabinFilters', component: CabinFiltersPage },
        { path: '/brake_pads', name: 'brakePads', component: brakePads },
```

```
        { path: '/brake_fluids', name: 'brakeFluids', component: brakeFluids },
        { path: '/engine_oils', name: 'engineOils', component: engineOils },
        { path: '/oil_filters', name: 'oilFilters', component: oilFilters },
        { path: '/fuel_filters', name: 'fuelFilters', component: fuelFilters },
        { path: '/air_filters', name: 'airFilters', component: airFilters },
        { path: '/coolants', name: 'coolants', component: coolants },
        { path: '/transmission_oils', name: 'transmissionOils', component:
transmissionOils },
        { path: '/company_information', component: companyInformation },
    ]
});

router.beforeEach((to, from, next) => {
    window.scrollTo(0, 0);
    next();
});

export default router;
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Вебзастосунок для пошуку автозапчастин

Програма та методика тестування

КПІ.ІТ04-9426.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:
_____ Павло РАДІОНОВ

Виконавець:
_____ Василь СТУКАНЕЦЬ

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ 3
2 МЕТА ТЕСТУВАННЯ 4
3 МЕТОДИ ТЕСТУВАННЯ 5
4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ 5

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є сайт для пошуку автозапчастин DeCar. А саме його функціональність

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– знаходження проблем, помилок і недоліків з метою їх усунення;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– системне тестування – перевіряється усе програмне забезпечення в цілому;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– тестування інтерфейсу користувача;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– динамічне тестування на відповідність функціональним вимогам;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Вебзастосунок для пошуку автозапчастин

Керівництво користувача

КПІ.ІТ04-9426.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:
_____ Павло РАДІОНОВ

Виконавець:
_____ Василь СТУКАНЕЦЬ

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ 3
2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ..... 4
2.1 Системні вимоги для коректної роботи..... 4
2.2 Завантаження застосунку..... 4
2.3 Перевірка коректної роботи..... 4
3 ВИКОНАННЯ ПРОГРАМИ 5

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

DeCar - це веб-застосунок, розроблений для спрощення процесу пошуку, порівняння та придбання автомобільних запчастин.

Застосунок забезпечує користувачам можливість швидко знайти необхідні запчастини для їх автомобілів, використовуючи зручний і інтуїтивно зрозумілий інтерфейс. Користувачі можуть шукати запчастини за маркою, моделлю.

Опис кінцевої збірки програмного забезпечення:

Інтерфейс користувача: Веб-сторінки, розроблені з використанням Vue.js, що забезпечують швидкий та інтерактивний досвід для користувачів.

Серверна частина: Серверна логіка реалізована на базі Django, що дозволяє ефективно обробляти запити та взаємодіяти з базою даних.

База даних: Використовується PostgreSQL для зберігання даних про автомобілі, запчастини та користувачів.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність мобільного пристрою з операційною системою на базі Android з версією 6.0 або вище. Або мобільного пристрою з ОС на базі iOS з версією 7.0 або вище або будь який інший персоональний комп’ютер який має підтримку HTTPS

– наявність доступу до Інтернету;

2.2 Завантаження застосунку

Не потребується

2.3 Перевірка коректної роботи

По завершенню завантаження сайту, повинна відкритись головна сторінка сайту. Якщо сторінка не відобразиласть, значить наявні проблеми або у користувача або зі сторони серверу.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску прогрми користувач бачить головну сторінку застосунку, та різні вкладки.

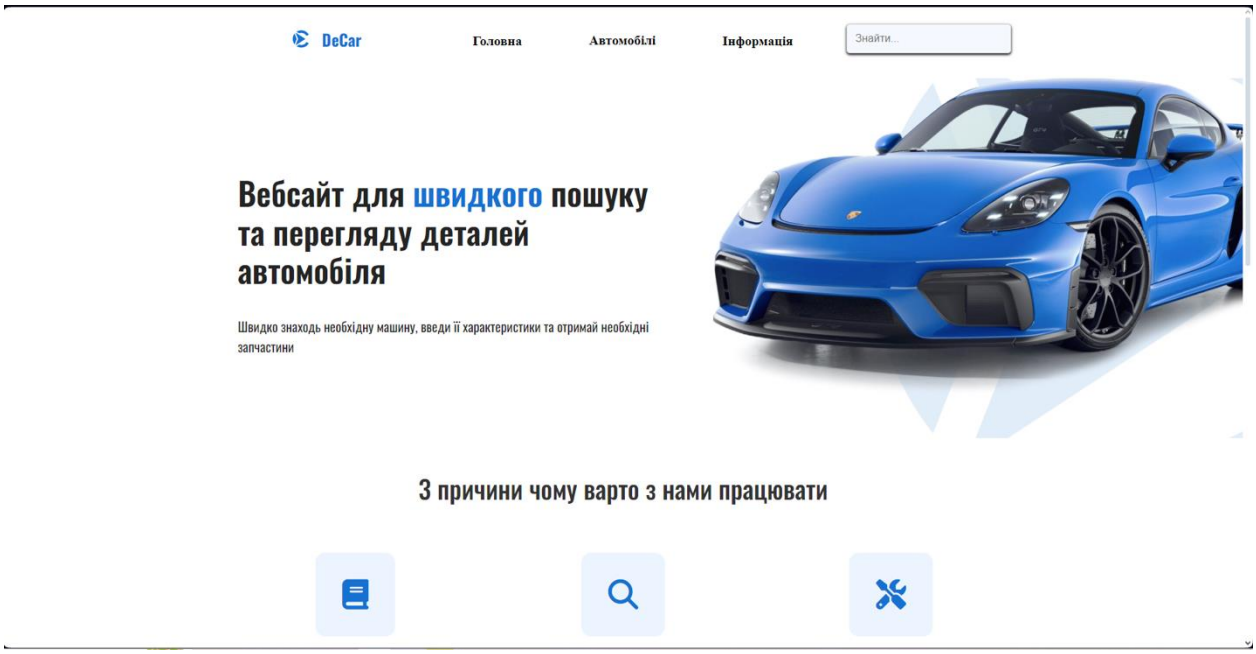

Рисунок 4.2 – Головна сторінка сайту

Для початку спробуємо пошукати автомобіль за моделлю. Для цього натискаємо на поле пошуку та вводимо назву моделі Octavia. Як можна побачити, на рисунку 4.3 система успішно провела пошук моделі в базі даних та запропонувала користувачу варіант.

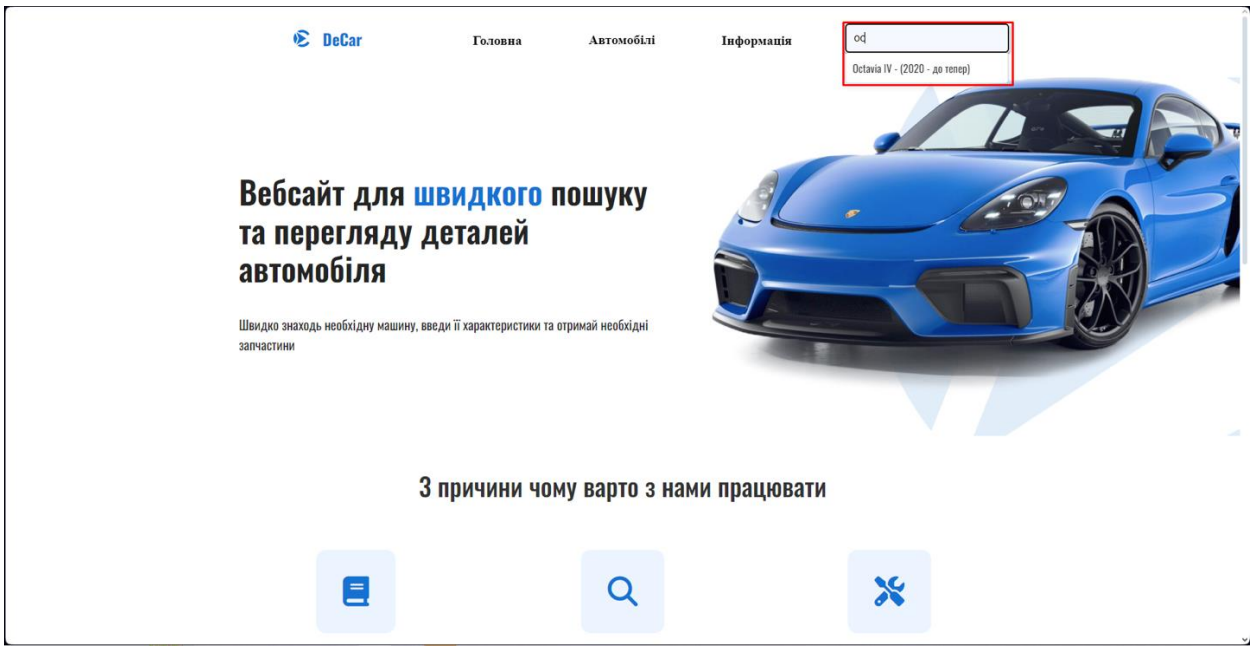

Рисунок 4.3 – Зовнішній вигляд сайту при спробі пошуку

Перейдемо до пошуку за маркою авто. Для цього переходимо на вкладку Автомобілі, та обираємо Volkswagen. Як видно на рисунку 4.4, система успішно відобразила всі доступні моделі авто марки Volkswagen.

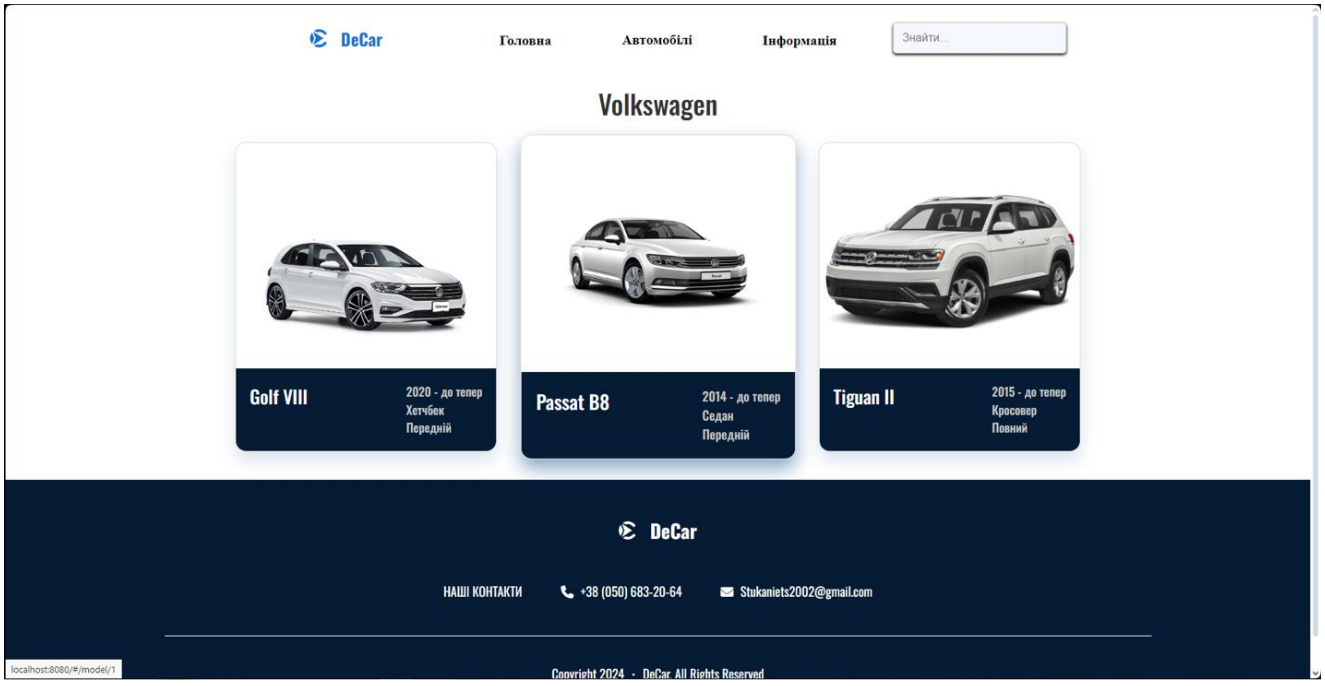

Рисунок 4.4 – зовнішній вигляд сайту при пошуку авто за маркою

Далі обираємо модель яка нас цікавить, а саме, Golf. Як видно на рисунку 4.5, система успішно перевела користувача на сторінку автомобіля відобразила основну інформацію про нього..

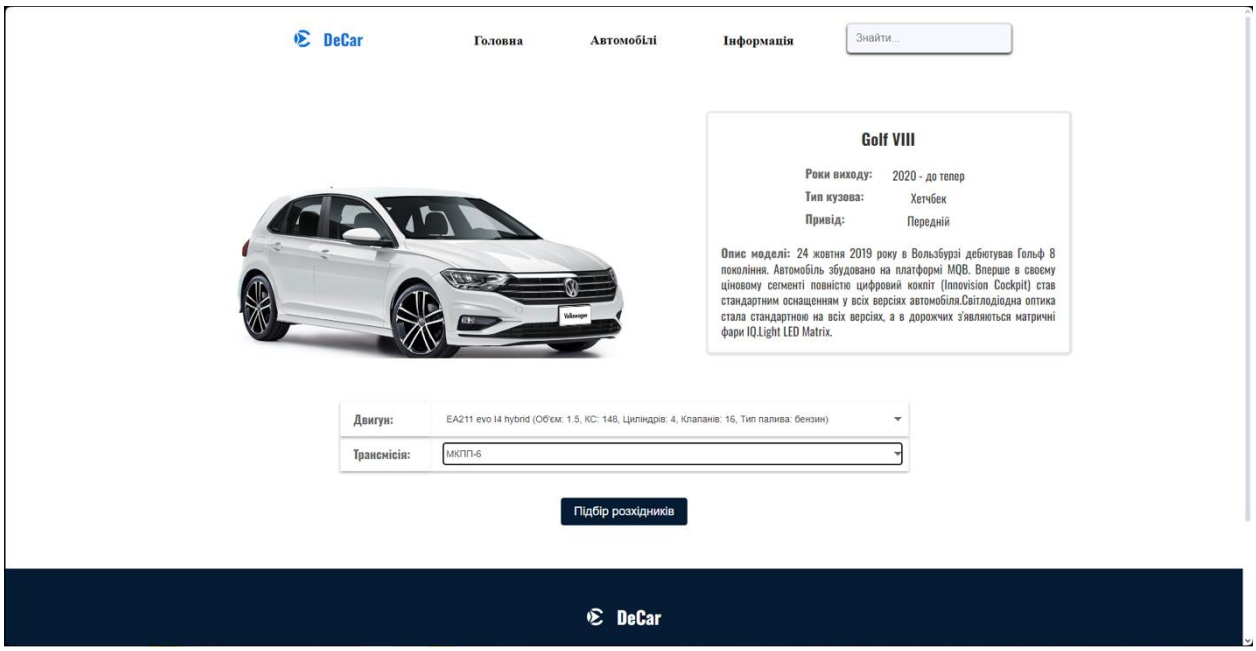

Рисунок 4.5 – Зовнішній вигляд сайту при переході на сторінкку автомобіля

Далі протестуємо поведінку сайту при виборі двигуна авто. Для цього на стрінці з інформацією про автомобіль на випадаючий список під назвою двигун та, як видно на рисунку 4.6, бачимо перед собою список доступних двигунів для даного автомобіля.

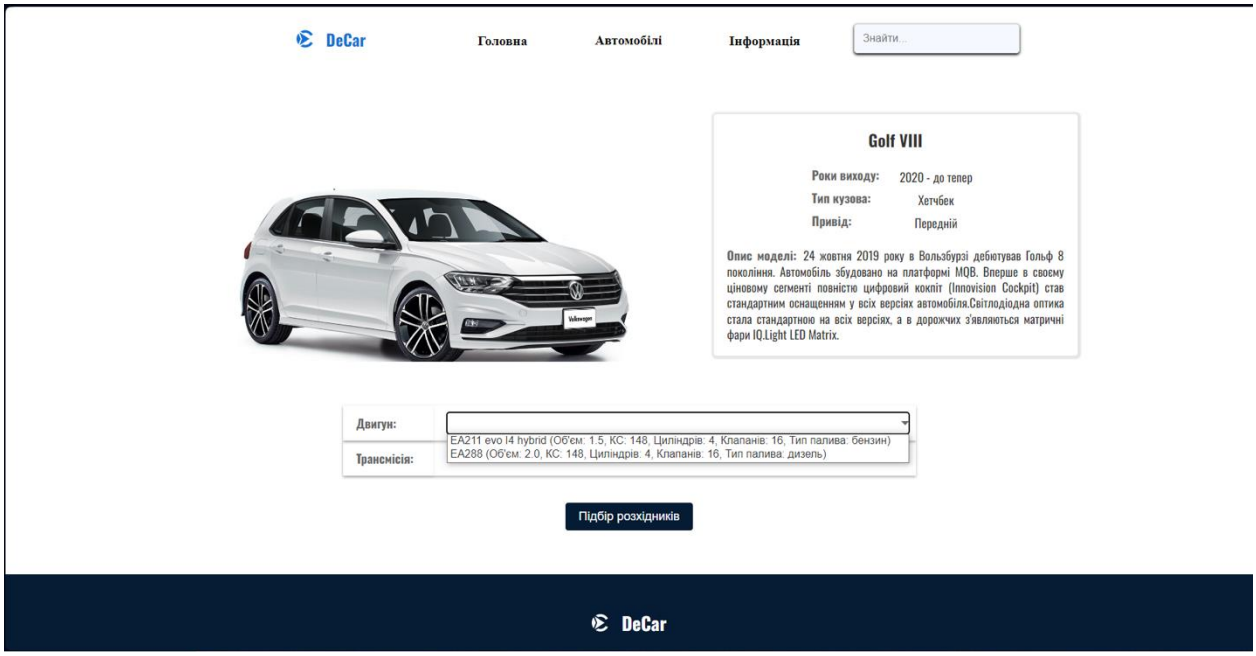

Рисунок 4.6 - Зовнішній вигляд сайту при виборі двигуна

Далі протестуємо поведінку сайту при виборі трансмісії авто. Для цього на сторінці з інформацією про автомобіль на випадаючий список під

назвою трансмісія та, як видно на рисунку 4.7, бачимо перед собою список доступних двигунів для даного автомобіля.

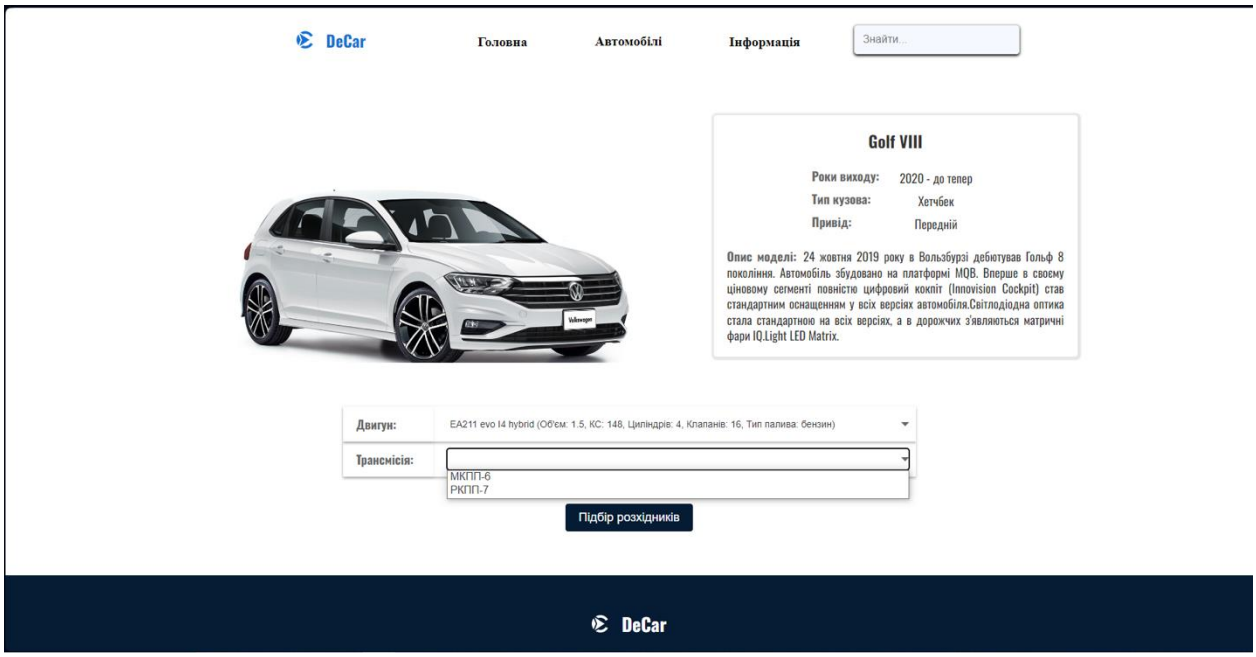

Рисунок 4.7 - Зовнішній вигляд сайту при виборі трансмісії

Після того як було обрано двигун та траснмісію, натискаємо на кнопку підбору розхідників. Як видно на рисунку 4.8, застосунок успішно перевів користувача на сторінку з доступними категоріями запчастин та відобразив їх.

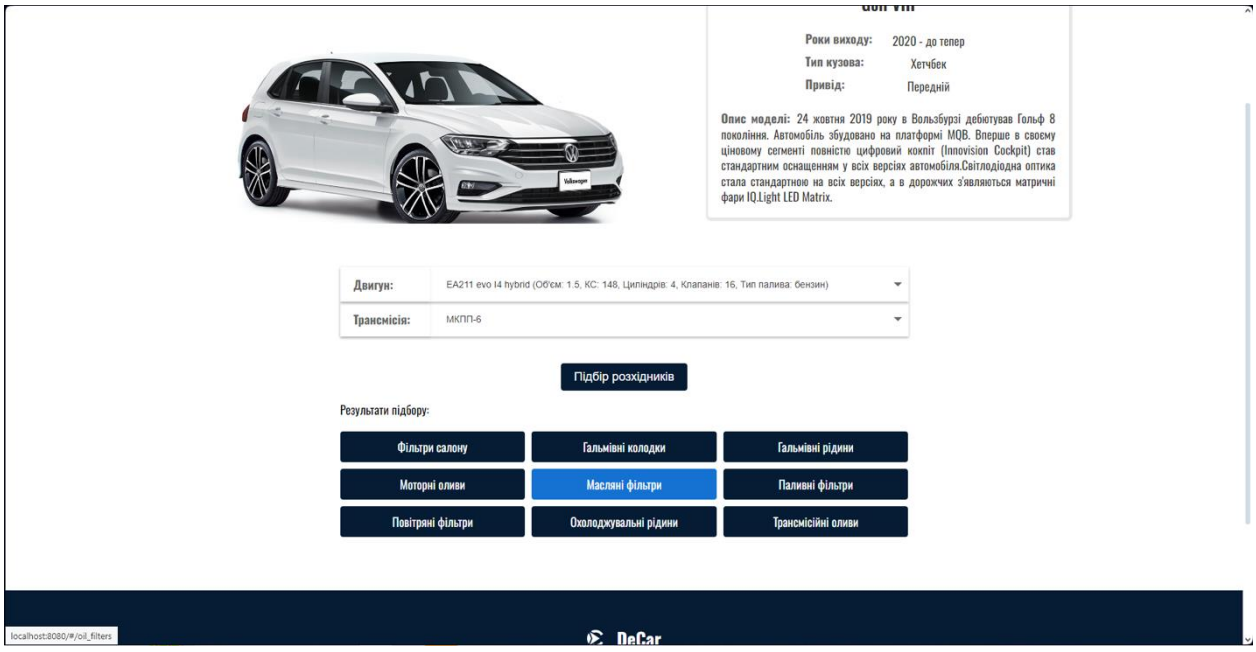

Рисунок 4.8 – Зовнішній вигляд сайту після підбору розхідників

Далі обираємо категорію «Масляні фільтри» та як бачимо на рисунку 4.9, застосунок успішно переводить користувача на сторінку з доступними запчастинами в даній категорії та їх описом.

Рисунок 4.9 – Зовнішній вигляд сторінки з доступними розхідниками

Далі перейдемо на вкладку з інформацією про компанію. Для цього в меню зверху обираємо вкладку «Інформація». Як видно на рисунку 4.10, застосунок успішно перевів нас на сторінку з інформацією про компанію.

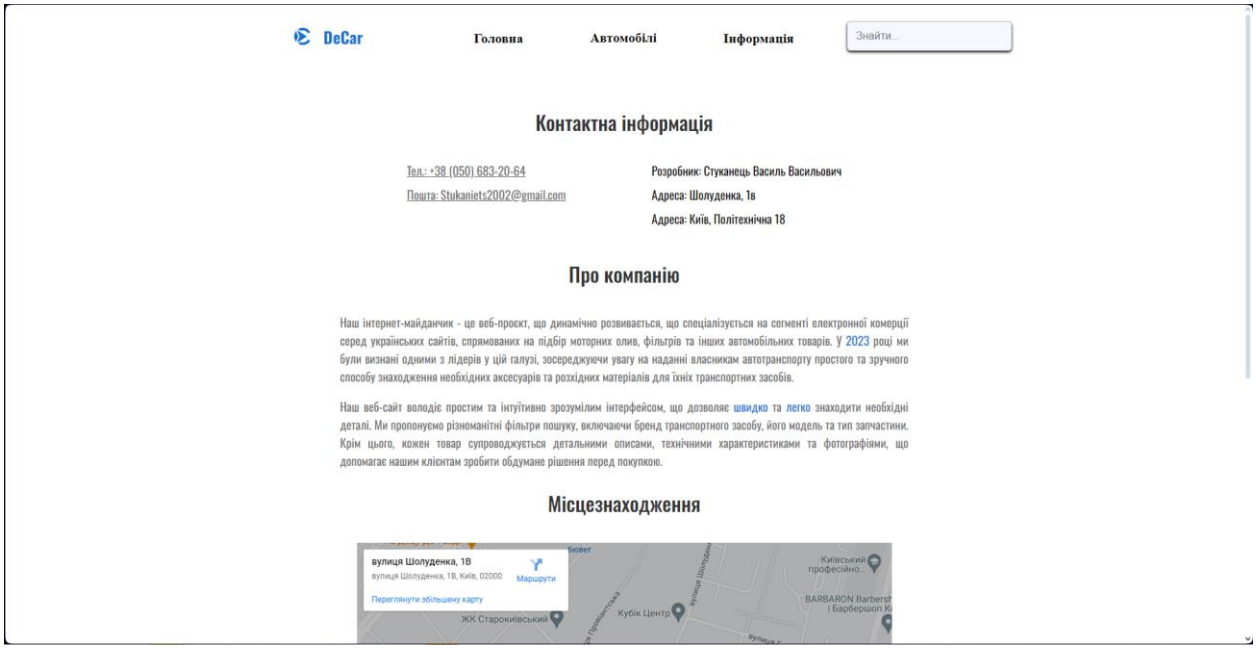

Рисунок 4.10 - Зовнішній вигляд сторінки з інформацією про компанію

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Вебзастосунок для пошуку автозапчастин

Графічний матеріал

КПІ.ІТ04-9426.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:
_____ Павло РАДІОНОВ

Виконавець:
_____ Василь СТУКАНЕЦЬ

Київ – 2024

РЕЦЕНЗІЯ

на дипломний проєкт

на здобуття ступеня бакалавра,

виконаний на тему: **«Вебзастосунок для пошуку автозапчастин»**

студентом (-кою) **Стуканцем Василем Васильовичем**

Поданий на рецензування дипломний проєкт містить 142 аркуші тексту, 22 ілюстрації, 28 таблиць, 7 джерел.

Робота повністю відповідає затвердженій темі та є актуальною.

Підставою для розробки вебзастосунку для пошуку автозапчастин є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Перевагами вважаю те, що дипломний проєкт вирішує нагальну проблему та пропонує зручне рішення для покращення процесу пошуку та придбання автозапчастин. Проєкт підкреслює важливість простоти використання, що сприяє задоволенню користувачів та отриманню позитивних результатів.

До недоліків можна віднести відсутність реалізації мобільної версії вебзастосунку, що може обмежувати зручність використання для користувачів, які віддають перевагу мобільним пристроям.

В цілому дипломний проєкт відповідає вимогам університету до дипломних проєктів. Вважаю, що дипломний проєкт заслуговує оцінки «відмінно», а його автор - студент Стуканець Василь Васильович заслуговує присудження ступеня бакалавра зі спеціальності «121 Інженерія програмного забезпечення» та кваліфікації бакалавр з інженерії програмного забезпечення.

Рецензент

посада, науковий ступінь, вчене звання _____ Ім’я ПРІЗВИЩЕ

ВІДГУК

керівника дипломного проєкту
на здобуття ступеня бакалавра,
виконаного на тему: **«Вебзастосунок для пошуку автозапчастин»**
студентом (-кою) **Стуканець Василь Васильович**

Дипломний проєкт присвячений розробці та впровадженню вебзастосунку для пошуку автозапчастин - DeCar.

Розроблений вебзастосунок допомагає користувачам спростити процес пошуку автомобільних запчастин за допомогою зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє швидко знайти потрібні запчастини за різними критеріями. Платформа також забезпечує користувачам можливість порівнювати ціни та характеристики запчастин, що робить процес вибору більш обґрунтованим та ефективним.

Підставою для розробки вебзастосунку є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Пояснювальна записка складається з п’яти розділів: Аналіз вимог до програмного забезпечення, Розроблення вимог до програмного забезпечення, Моделювання та конструювання програмного забезпечення, Аналіз якості та тестування програмного забезпечення, Впровадження та супровід програмного забезпечення. У розділі загальних положень описано предметне середовище, постановку задачі та огляд наявних аналогів. У розділі моделювання та конструювання програмного забезпечення було описано інструменти, за допомогою яких проведено практичну реалізацію застосунку, та описано модулі застосунку та їх методи. В другому розділі було описано вимоги до функціональної частини програмного забезпечення. В третьому розділі було розроблено та описано архітектуру програмного забезпечення. В четвертому розділі було проведено тестування додатку, включаючи тестування інтерфейсу користувача. П’ятий розділ описує процес розгортання вебзастосунку.

В цілому дипломний проєкт відповідає вимогам університету до дипломних проєктів. Вважаю, що студент Стуканець Василь Васильович заслуговує присудження ступеня бакалавра зі спеціальності «121 Інженерія програмного забезпечення» та кваліфікації бакалавр з інженерії програмного забезпечення.

Керівник дипломного проєкту
асистент _____ Владислав Сарнацький

КПІ.ІП04-9426.045440.06.99.ССВ

					КПІ.ІП04-9426.045440.06.99.ССВ	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використаннь	
Розробив		Стуканець В.В				Аркуш / Аркушів
Перевірив		Сарнацький В.В				
Т. кон.						
Н. кон.		Радіонов П.Ю			Вебзастосунок для пошуку автозапчастин	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-04
Затвердив		Жаріков Е.В				

КПІ.ІП04-9426.045440.06.99.СБД

					КПІ.ІП04-9426.045440.06.99.СБД
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних
Розробив		Стуканець В.В			Літера: — Маса: — Масштаб: —
Перевірив		Сарнацький В.В			Аркуш: — Аркушів: —
Т. кон.					
Н. кон.		Радіонов П.Ю			Вебзастосунок для пошуку автозапчастин
Затвердив		Жаріков Е.В			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-04

					КПІ.ІП04-9426.045440.06.99.КЕ
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм
Розробив		Стуканець В.В			
Перевірив		Сарнацький В.В			
Т. кон.					
Н. кон.		Радіонов П.Ю			Вебзастосунок для пошуку автозапчастин
Затвердив		Жаріков Е.В			

Літера	Маса	Масштаб
Аркуш	Аркушів	

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-04