

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Веб - сервіс об'яв з продажу авто з можливістю
порівняння ціни»**

Виконав студент групи ІПЗм-21-2
ПЕРШКО Владислав Сергійович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
 ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
 ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
 КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
 програмного забезпечення

Андрій МОРОЗОВ

«2» лютого 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **ПЕРШКО Владислав Сергійович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Веб-сервіс об'яв з продажу авто з можливістю порівняння ціни»,**

затверджена Наказом закладу вищої освіти від «2» лютого 2023 р. № 57/с

Вихідні дані для роботи: предметом дослідження є методи та засоби розробки веб - сервісу об'яв з продажу авто. Об'єктом дослідження є побудова моделей для веб - сервісу об'яв з продажу авто.

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	<i>Кравченко С.М.</i>	02.02.2023р.	12.03.2023р.
2	<i>Кравченко С.М.</i>	02.02.2023р.	12.03.2023р.
3	<i>Кравченко С.М.</i>	02.02.2023р.	12.03.2023р.

Дата видачі завдання: «2» лютого 2023 р.

РЕФЕРАТ

Пояснювальна записка до магістерської роботи містить 60 сторінок, 32 ілюстрацій та 19 таблиць.

Кваліфікаційна магістерська робота полягає в науковому дослідженні та аналізі методів і моделей веб-сервісів об'яв з продажу авто.

Робота на тему «Веб – сервіс об'яв з продажу авто з можливістю порівняння ціни» складається з теоретичного та програмного комплексу системи і пояснювальної записки.

В роботі визначено основні завдання на веб - сервісу, проаналізовано існуючі технології, обґрунтовано вибір технологій.

КЛЮЧОВІ СЛОВА: Веб - сервіс, Розробка, Аналіз, Дослідження.

ABSTARCT

The explanatory note to the master's thesis contains 60 pages, 32 illustrations and 19 tables.

The qualifying master's work consists of scientific research and analysis of methods and models of web services for car sales ads.

The work on the topic "Web - a service of ads for the sale of cars with the possibility of price comparison" consists of a theoretical and software complex of the system and an explanatory note..

The work defines the main tasks of the web service, analyzes existing technologies, and substantiates the choice of technologies.

KEY WORDS: Web service, Development, Analysis, Research.

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ І ДОСЛІДЖЕННЯ АВТОМОБІЛЬНИХ РИНКІВ.....	8
<i>1.1 Основні поняття про автомобільний ринок України</i>	<i>8</i>
<i>1.2 Аналіз та дослідження автомобільного ринку України.....</i>	<i>9</i>
<i>1.3 Аналіз існуючих аналогів програмного продукту.....</i>	<i>13</i>
<i>1.4 Розробка архітектури додатку.....</i>	<i>21</i>
<i>1.5 Обґрунтування вибору технологій</i>	<i>22</i>
<i>Висновок до першого розділу.....</i>	<i>25</i>
РОЗДІЛ 2. МЕТОДИ ТА АРХІТЕКТУРА ВЕБ – СЕРВІСУ ОБ’ЯВ З ПРОДАЖУ АВТО З МОЖЛИВІСТЮ ПОРІВНЯННЯ ЦІНИ.....	26
<i>2.1 Методи та аналіз сервісу порівняння цін</i>	<i>26</i>
<i>2.2 Моделювання процесів веб – додатку.....</i>	<i>30</i>
<i>2.3 Розробка бази даних для веб – додатку.....</i>	<i>31</i>
<i>2.4 Проектування та реалізація алгоритмів веб – додатку</i>	<i>39</i>
<i>2.5 Проектування та реалізація алгоритмів веб – додатку</i>	<i>46</i>
<i>Висновок до другого розділу.....</i>	<i>53</i>
РОЗДІЛ 3. ІНТЕРФЕЙС ТА АРХІТЕКТУРА ВЕБ - ДОДАТКУ	55
<i>3.1 Архітектура та системні вимоги до веб – додатку.....</i>	<i>55</i>
<i>3.2 Структура інтерфейсу та порядок роботи з веб - сервісом.....</i>	<i>56</i>
<i>3.3 Тестування роботи веб – додатку.....</i>	<i>60</i>
<i>Висновок до третього розділу</i>	<i>61</i>
ВИСНОВКИ	62
ПЕРЕЛІК ВИКОРСИТАНОЇ ЛІТЕРАТУРИ	63
ДОДАТКИ.....	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	– База даних.
JS	– JavaScript
UI	– User Interface
API	– Application Programming Interface

ВСТУП

Актуальність теми. В сучасному світі існує багато веб - сервісів об'яв з продажу авто і вже не обов'язково їздити безпосередньо на авто базар для того щоб підібрати собі машину, зараз достатньо відкрити браузер у смартфоні і знайти те авто, яке хочеш і в пару кліків дізнатися всю інформацію про авто.

Але для того щоб знайти собі авто, потрібно заходити на різні сервіси, на кожному з них реєструватися і вручну порівнювати ціну, тому що немає сервісу, який би збирав всю інформацію про різні об'яви з продажу авто на різних сервісах.

Метою магістерської роботи є аналіз розробки та дослідження життєздатності проекту веб – сервісу об'яв з продажу авто з можливістю порівняння ціни.

Постановлена мета визначає наступні цілі:

- проведення аналізу розробки проекту;
- дослідження життєздатності проекту;
- розробка проекту з використанням сучасних технологій.

Об'єктом дослідження є методи та засоби розробки веб - сервісу об'яв з продажу авто з можливістю порівняння ціни.

Предмет дослідження: побудова моделей для веб - сервісу об'яв з продажу авто з використанням сучасних інформаційних технологій.

Практичне завдання отриманих результатів: розробка веб – сервісу об'яв з продажу авто з використання сучасних технологій для україномовних користувачів.

РОЗДІЛ 1. АНАЛІЗ І ДОСЛІДЖЕННЯ АВТОМОБІЛЬНИХ РИНКІВ

1.1 Основні поняття про автомобільний ринок України

Автомобільний ринок вважається однією з найбільш динамічно розвинутих галузей промисловості української економіки.

Незважаючи на те, що український автомобільний ринок став одним з основних секторів економіки країни за останнє десятиліття, кількість досліджень, присвячених українському ринку легкових автомобілів, все ще дуже обмежена. Досліджень, присвячених українському ринку легкових автомобілів недостатньо і більшість з них обмежуються оглядом ринку та обговоренням діяльності кожного конкретного автовиробника та автодилера.

Автомобіль забезпечує високу мобільність, ефективність роботи та визначає сучасний стиль життя суспільства. Він є показником рівня матеріального добробуту як окремих осіб (їх власників), так і суспільства або країни в цілому (в даному випадку мова йде про рівень забезпеченості населення країни автомобілями).

Останніми роками дилерські мережі набули значного поширення на українському автомобільному ринку з таких причин, як підвищення ефективності продажів автомобілів, висока конкурентоспроможність, зниження витрат обігу на одиницю продукції, більш швидке реагування на зміну ринкової кон'юнктури та впровадження сучасних технологій продажів. Однак, через специфіку своєї діяльності, дилери мають багато особливостей у своєму функціонуванні та розвитку.

Формування дилерської мережі є основним стратегічним напрямком розвитку процесу продажу автомобілів. Тип організації торгівлі, створений на базі мережі дилерських підприємств, має дві важливі особливості, які суттєво підвищують конкурентоспроможність цієї форми організації торгівлі на сучасному ринку. Концентрація комерційних функцій за рахунок ефекту масштабу знижує витрати на організацію технічних процесів, а завдяки

єдиному управлінню всіма підприємствами дилерської мережі дає можливість забезпечити сильний опір конкурентам.

Автомобіль – це вид транспортного засобу, який належить до групи товарів довготривалого використання. На попит цього транспортного засобу впливають наступні фактори:

- **Об'єктивні:**

- Дохід населення
- Курс обміну іноземних валют
- Інфляція та зростання цін
- Ставки в банках за депозитами та кредитами

- **Суб'єктивні:**

- Поточне сприйняття ситуації споживачами
- Надія та очікування змін у майбутньому
- ІСН – це загальний показник, який отримується в результаті досліджень та опитувань споживачів, що базується на:
 - Оцінці поточної фінансової ситуації
 - Оцінці змін у фінансовій ситуації
 - Оцінці розвитку в країні в найближчі роки

1.2 Аналіз та дослідження автомобільного ринку України

Останніми роками дилерські мережі набули значного поширення на українському автомобільному ринку з таких причин, як підвищення ефективності продажів автомобілів, висока конкурентоспроможність, зниження витрат обігу на одиницю проданої продукції, швидке реагування на зміну ринкової кон'юнктури та впровадження сучасних методів продажу. Однак через специфіку своєї діяльності дилерські мережі мають певні особливості у своєму функціонуванні та розвитку.

Конкуренція на автомобільному ринку відбувається не тільки на рівні виробників і дистриб'юторів, але й на локальному ринку, який характеризується наявністю декількох дилерів одного бренду. Це впливає на методи та інструменти просування бренду на локальному ринку. Адже головна мета виробника в маркетингових комунікаціях - створити імідж бренду, представити його цінності та сформувати лояльність і прихильність, тоді як дилери на локальному ринку залучають певних потенційних покупців до конкретного автосалону. Мета полягає в тому, щоб зробити саме це. При цьому виробники, прагнуть збільшити продажі, а отже, і прибутки.

За даними онлайн ресурсу planetavto.ua був згенерований наступний графік:

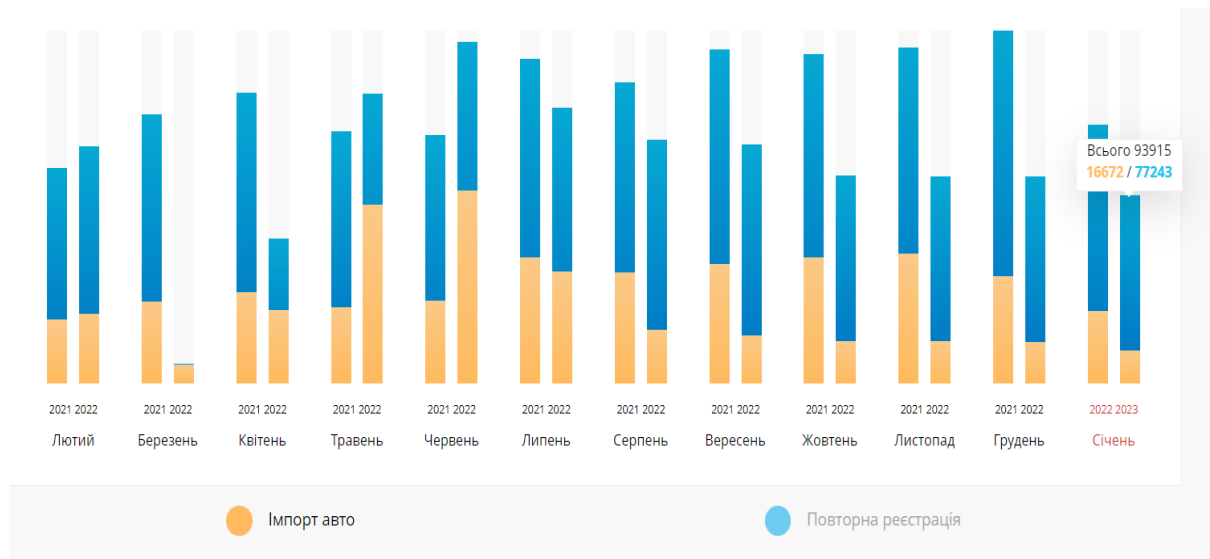


Рис. 1.1 – Статистика продажу Б/У авто за 2022 – 2023

Онлайн-попит на автомобілі в Україні зростає вже кілька років поспіль. При цьому темпи зростання залишаються стабільно високими - понад 23% на рік. А введення карантинних умов вже навесні 2020 року також сприяло зростанню інтересу до цієї теми. Це створює умови для того, щоб поточна тенденція збереглася і в майбутньому.

Веб – сервіси об'яв все ще формують значну частину попиту на авторинку і створюють його специфіку. В Україні створена асоціація "питання з авто - агрегатор". Про це свідчить висока частка онлайн-попиту на такі сайти - 41,5% від загальної кількості.

За три роки, з липня 2018 по червень 2021, онлайн - попит на автомобілі зріс у 1,5 рази. Це один з найбільших онлайн - ринків в Україні. А карантинні обмеження лише збільшили інтерес українців до цієї теми: за останній рік з липня 2020 по червень 2021 року тему автомобілів шукали в мережі інтернет близько 472 мільйонів разів.

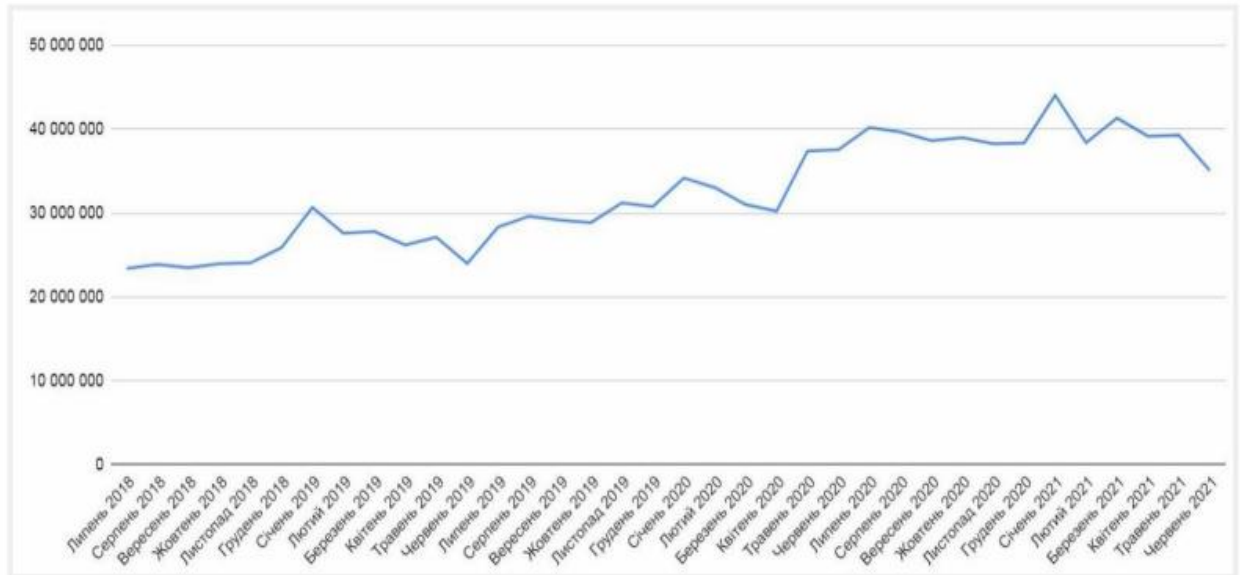


Рис. 1.2 – Онлайн попит на авто в Україні 2018 – 2021

З графіку вище (Рис 1.2) чітко видно майже стабільне зростання попиту онлайн – ринку авто досить великими темпами.

Із запровадженням карантинних обмежень громадяни України почали активно моніторити ціни на автомобілі онлайн. З квітня 2020 року по серпень 2020 року їх кількість зросла в 1,7 рази (в середньому з 200 000 до 3,47 млн випадків на місяць). І ця тенденція не збирається зменшуватися.

До пандемії попит на продаж автомобілів був циклічним, з піками інтересу наприкінці осені та на початку зими, але з 2020 року цей попит втратив свою циклічність. А сам попит за останні три роки впав на 9%.

Динаміка попиту на оренду автомобілів є незмінною і сезонною, з піками в літні місяці. В середньому, за послугою оренди автомобілів звертаються понад 120 000 разів на місяць, що становить 3% від загального комерційного попиту.

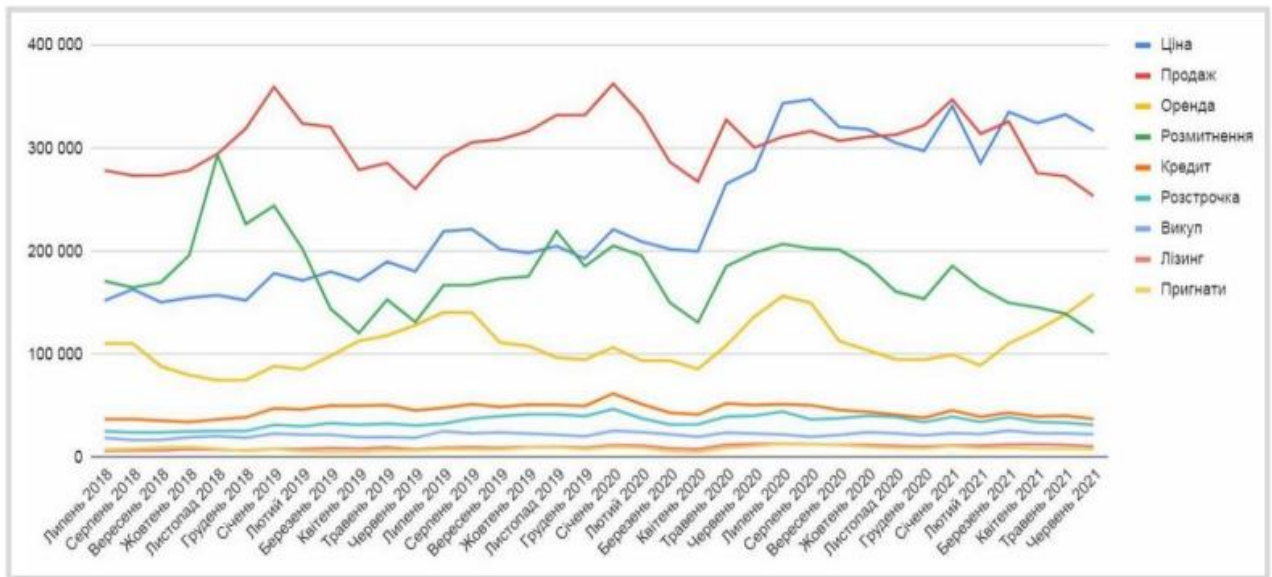


Рис. 1.3 – Онлайн попит на авто без купівлі в Україні 2018 – 2021

Українці звикли шукати автомобілі в інтернеті через спеціалізовані платформи. Щомісяця понад 16,2 млн користувачів вирішують свої проблеми за схемою "потрібен автомобіль = агрегатор". Таким чином, на агрегатори припадає 41,5% від загального онлайн-попиту на автомобілі.

Українці шукають вживані автомобілі онлайн понад 1,35 мільйона разів на місяць, що виводить цей показник безпосередньо в пошуковий рядок. Це становить 3,66% від загального попиту на автомобільному ринку або 16,2 млн. запитів на рік.

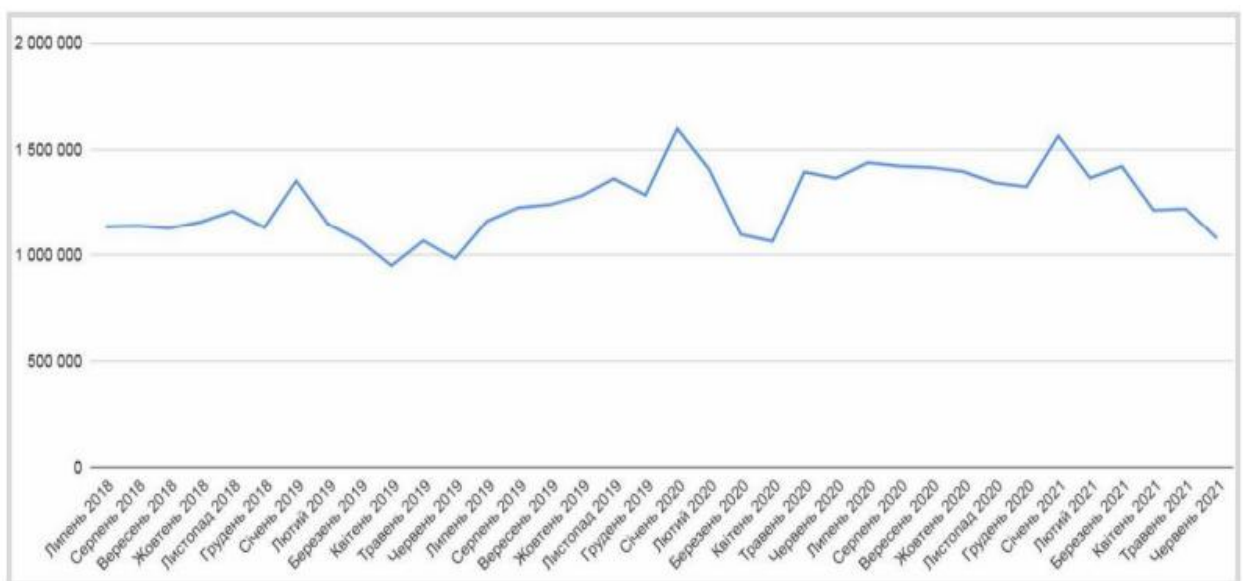


Рис. 1.4 – Онлайн попит на вживані авто в Україні 2018 – 2021

Українці шукають вживані автомобілі за різними типами запитів:

- Прямо від "вживаний" + різні фактори ("вживаний Nissan", "ціна вживаного Mercedes");
- Безпосередньо назви веб – сервісів об'яв з продажу вживаних авто ("autotia", "olx авто")

1.3 Аналіз існуючих аналогів програмного продукту

На сьогоднішній день найбільш популярними авторинками України вважаються «Auto.ria», «OLX авто», «RST», отже розглянемо кожен із цих веб – сервіс.

Першим розглянемо веб – сервіс «Auto.ria». З 2014 року працює як платформа для приватних осіб і компаній, які розміщують оголошення про купівлю та продаж нових і вживаних автомобілів. Компанія має офіси в Таллінні, Вінниці та Києві.

У жовтні 2016 року, за даними SimilarWeb, сайт посів 6 місце за відвідуваністю серед автомобільних сайтів у світі в категорії "авторинок" і 11 місце в категорії "автомобілі та транспорт".

Даний веб - сервіс публікує оголошення про продаж та характеристики автомобілів і працює як віртуальна автомобільна біржа.

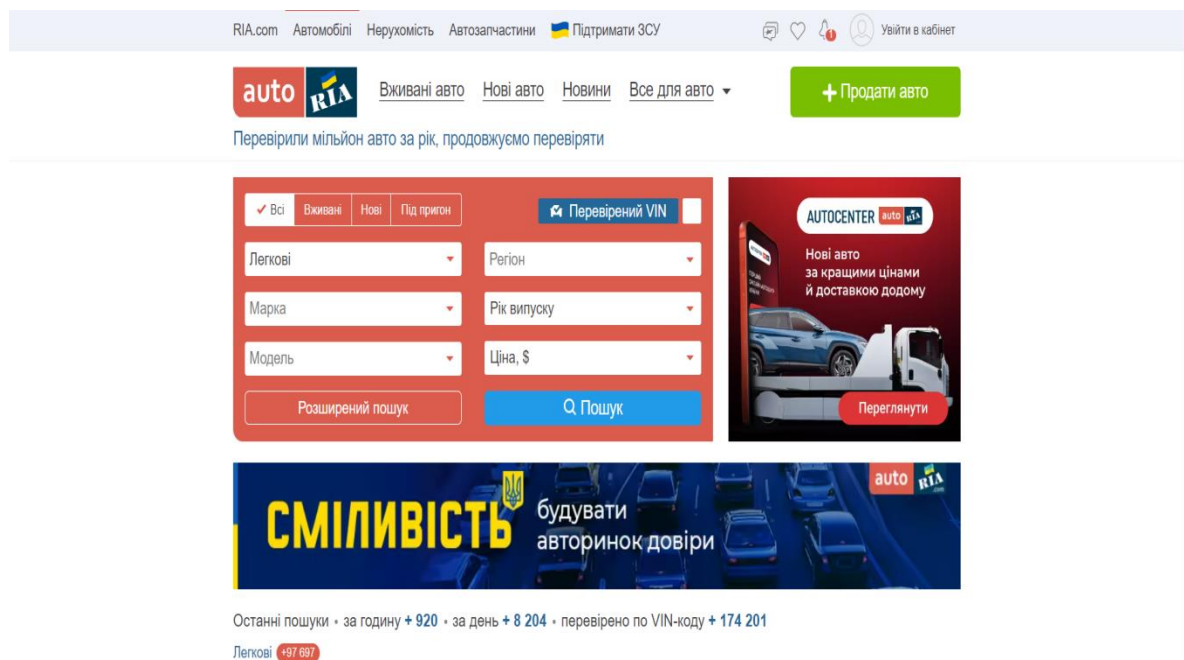


Рис. 1.5 – Головна сторінка «Auto.ria»

Головна сторінка даного веб – сайту має приємний вигляд, сучасний дизайн та гарну композицію кольорів. Одразу ж користувача зустрічає гнучкий фільтр пошуку, який можна, при бажанні зробити максимально детальним, щоб знайти те що потрібно.

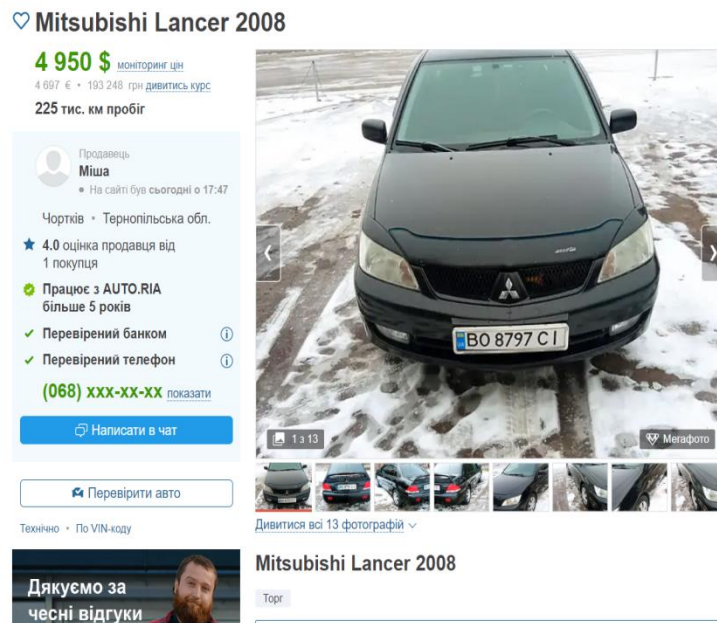


Рис. 1.6 – Сторінка об’яви «Auto.ria»

На сторінці об’яви користувач може побачити основну інформацію про власника автомобіля, фото автомобіля та базову інформацію про оголошення.

Перевірено AUTO.RIA за реєстрами MBC

BO 8797 CI

Перевірений VIN-код

JMBSNSCS3x8Uxxxx26

✓

Марка, модель, рік

Mitsubishi Lancer 2008

✓

Двигун

1.6 л • Газ / Бензин

✓

Колір

■ Чорний

✓

Остання операція

07.07.2020 • 2 роки, 7 міс. тому
перепродаж (на внутрішньому ринку)

✓

Кількість власників

5

✓

В розшуку

Ні

Перевірено AUTO.RIA за реєстрами рухомого майна

✓

Тип обтяження

не виявлено

✓

Обмеження відчуження

дозволено відчужувати

Перевірено AUTO.RIA по страховим базам

✓

Наявність страхового випадку

не знайдено

Перевірено AUTO.RIA за 15 базами даних

Пробіг перевірено

Даних про пробіг не виявлено
інформація відсутня в базах даних

Пробіг від продавця

225 тис.км

✓

ДТП

Немає офіційно зареєстрованих

Вам допомогли перевірки AUTO.RIA?

Подобається

Рис. 1.7 – Інформація про авто з об'яви «Auto.ria»

Нижчу на цій же сторінці користувач може ознайомитися з детальним описом авто. Штучний інтелект считує номерний знак авто з наявних в об'яві фото, та перевіряє дані про авто у відкритих реєстрах MBC, рухомого майна, страхових баз та перевіряє VIN – код авто.

Другий аналог є веб – сервіс «RST».

RST – чітко спеціалізований на розміщені оголошень з продажу авто. За даними сервісу «WhoIs» сайт був створений 3 квітня 2014 року і останній раз був оновлений 15 грудня 2022 року. Хоч і оновлення було нещодавно, інтерфейс вже застарів.

Найдешевші автомобілі продаються на «RST»; відсоток автомобілів вартістю до \$3 000 тут найвищий - 46%; також є невелика кількість

автомобілів вартістю понад \$15 - 3,1% від загальної кількості оголошень.
Середня ціна на цьому сайті - \$3 500.

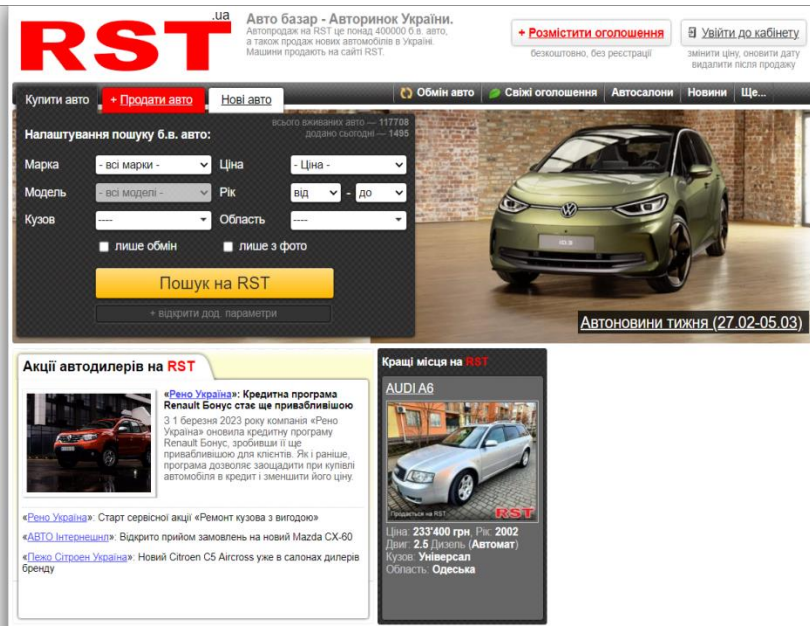


Рис. 1.8 – Головна сторінка «RST»

На головній сторінці, як і в попередньому випадку користувача зустрічає фільтр з можливістю розширеного пошуку. Також на головній є блок новин, та блок найкращих позицій серед інших об'яв.

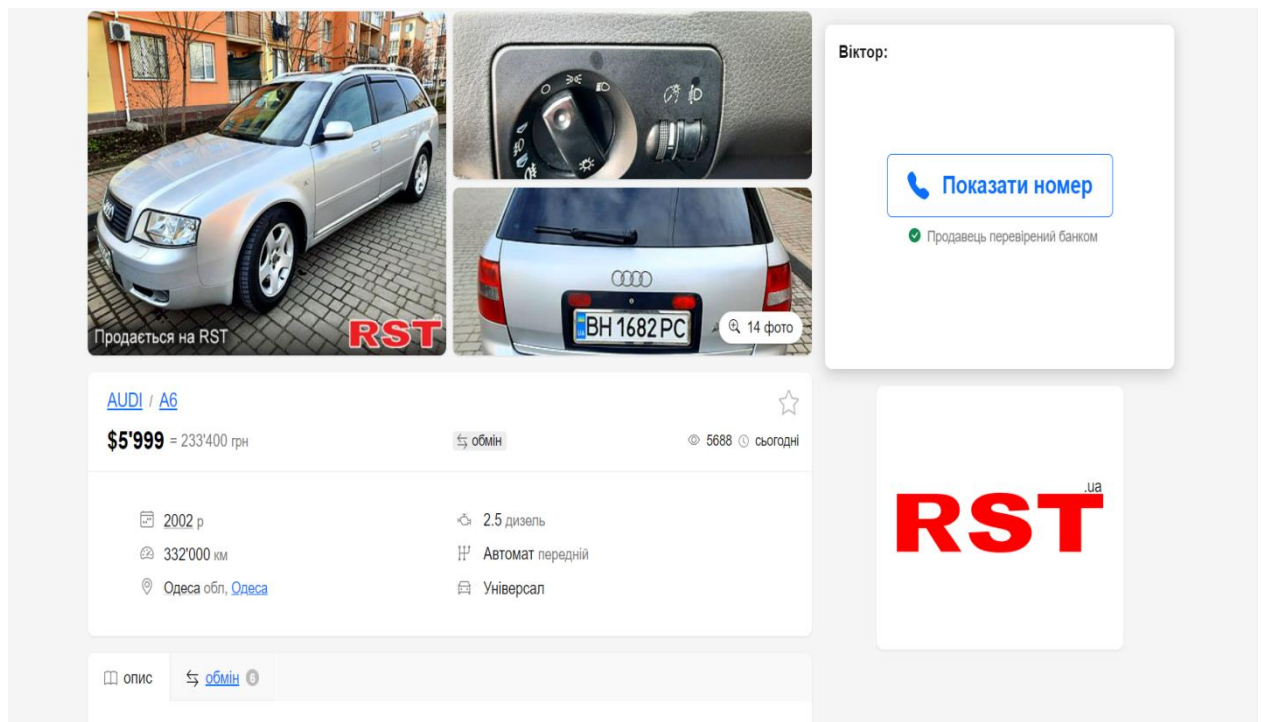


Рис. 1.9 – Сторінка об'яви «RST»

На сторінці об'яви даного веб – сервісу присутня основна інформація про авто, об'яви, короткий опис та базову інформацію про власника. Дана сторінка має приємний дизайн, веб – сторінка не перенасичена інформацією та легка для сприйняття користувачем, але, на мою думку, дуже мало інформації про авто.

Останнім аналогом є веб – сервіс «OLX авто».

OLX - це онлайн платформа оголошень, яка об'єднує людей, щоб купувати, продавати та обмінюватися товарами та послугами. Станом на 2018 рік на платформі зареєстровано 1,5 мільйона продавців, понад 11 мільйонів оголошень і близько 100 нових оголошень додаються щохвилини. Оголошення розміщені в таких категоріях, як Діти, Нерухомість, Транспорт, Автозапчастини, Робота, Тварини, Дім і сад, Електроніка, Бізнес та послуги, Мода і стиль, Хобі, Відпочинок і спорт, Вільний продаж та обмін OLX - найвідвідуваніший сервіс оголошень в Україні. Кожен другий український інтернет - користувач заходить на сайт принаймні раз на місяць.

За даними сервісу «WhoIs» сайт був створений 27 лютого 2013 року і останній раз був оновлений 17 січня 2022 року.

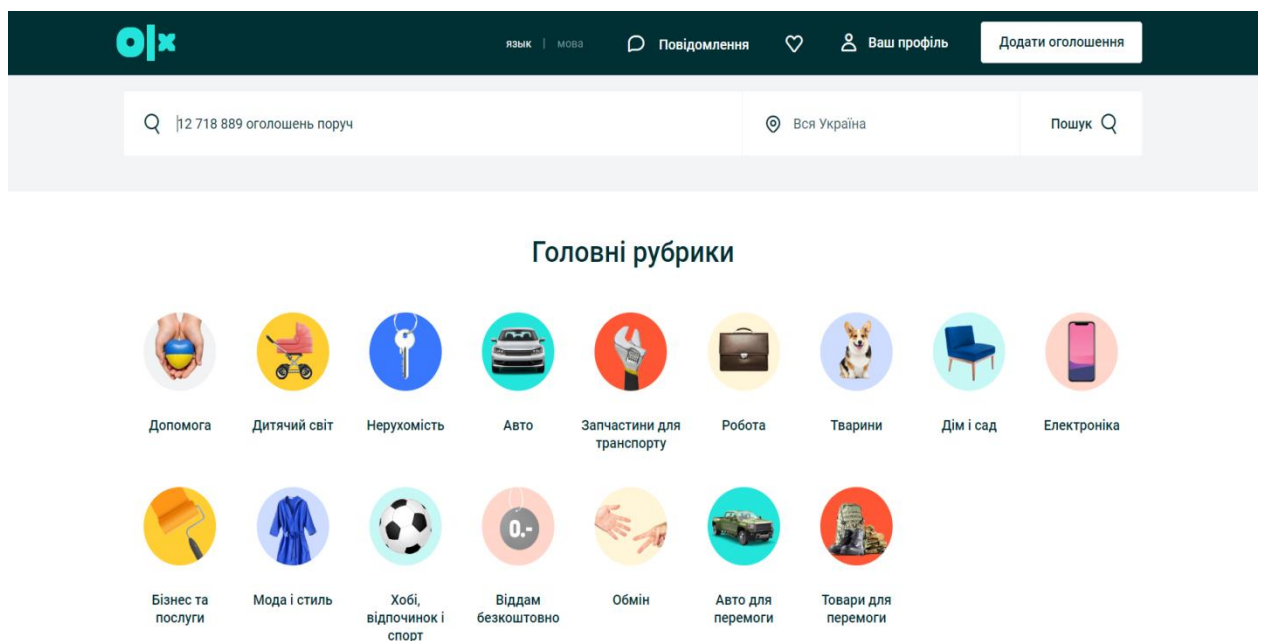


Рис. 1.10 – Головна сторінка «OLX»

Головна сторінка даного сервісу має мінімалістичний дизайн. На сторінці присутній пошук об'яв та вибір бажаної рубрики. Нас цікавить саме рубрика «Авто».

Фільтри

Категорія	Ціна	Тип автомобіля	Рік випуску	Умови продажу
Авто ▼	Від: до:	Всі оголошення ▼	Від: до:	Всі оголошення ▼
Пробіг тис.км.	Коробка передач	Вид палива	Тип кузова	Колір
Від: до:	Всі оголошення ▼	Всі оголошення ▼	Всі оголошення ▼	Всі оголошення ▼
Технічний стан	Розмитнена	Авто пригнано з		
Всі оголошення ▼	Всі оголошення ▼	Всі оголошення ▼		

Рис. 1.11 – Фільтр пошуку авто на «OLX»

Фільтр пошуку в даному веб – сервісі майже нічим не відрізняється від попередніх варіантів, окрім можливості розширеного пошуку, тут цієї можливості немає.

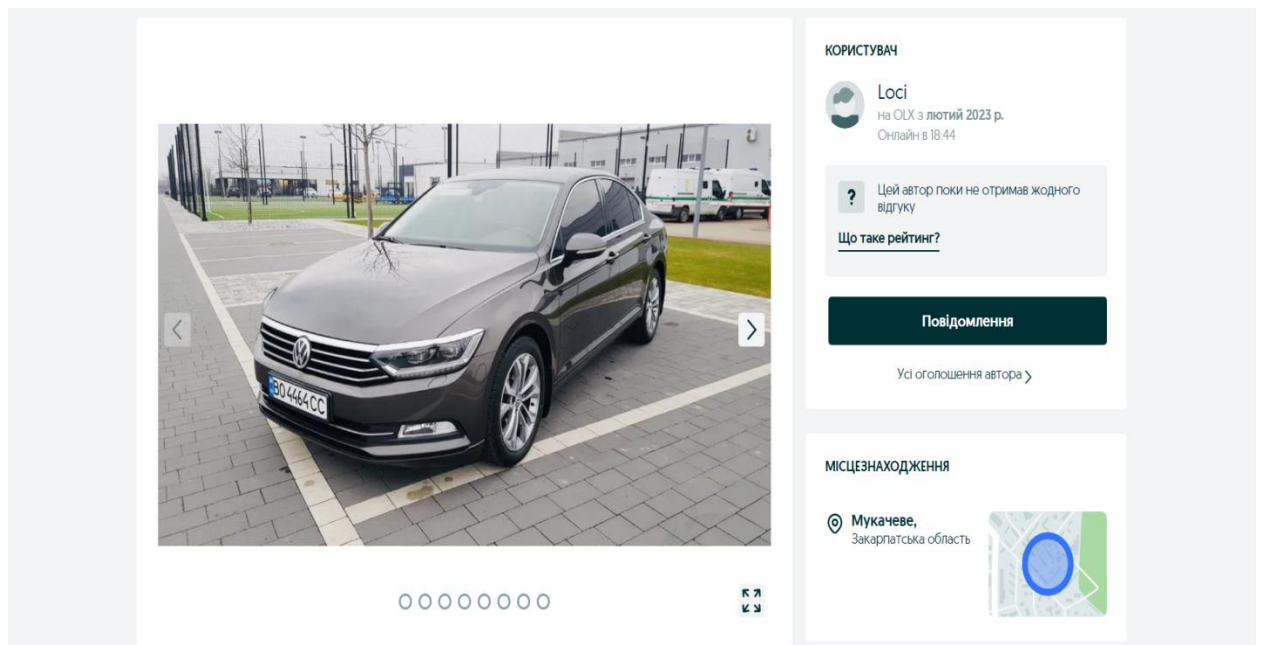


Рис. 1.12 – Сторінка об'яви «OLX»

На сторінці об'яви, як і в попередніх аналогах є фото авто, базова інформація про власника авто з можливістю написати йому. Внизу сторінки присутній короткий опис про авто та всі характеристики транспортного засобу.

У таблиці 1.1 наведено порівняльну характеристику існуючих аналогів.

Таблиця 1.1

Порівняльна характеристика існуючих аналогів

Порівняльні характеристики	Auto.ria	RST	OLX
Можливість реєстрації нового користувача	+	+	+
Розширений фільтр пошуку об'яв	+	+	-
Детальна інформація в оголошенні	+	-	-
Можливість написати в особисті повідомлення власнику авто	+	-	+
Можливість створення нових оголошень	+	+	+
Можливість редагування своїх об'яв	+	+	+
Зручний інтерфейс	+	-	+
Можливість безкоштовно	+	+	+

користуватись сервісом			
Можливість порівняння ціни	-	-	-

Кожен з проаналізованих ринків знайшов свою нішу. Оголошення, які публікують українці, відрізняються за ціною, роком реєстрації авто та регіоном продажу. На думку експертів Інституту автомобільного ринку, така ситуація може бути пов'язана з різною ринковою філософією та ціновою політикою ринків при розміщенні оголошень.

Наприклад, велика кількість "нових" і "дорогих" автомобілів на AUTO.RIA можна пояснити великою кількістю перевірок, які проходять продавці і автомобілі. Кожен автомобіль перевіряється за різними базами даних, які є корисними для потенційних покупців. Ці перевірки є більш важливими для нових і дорогих автомобілів, оскільки покупці готові ризикувати більшою сумою грошей.

Натомість немає сенсу тестувати автомобілі, випущені до 2000 року, які коштують кілька тисяч доларів, та й можливостей протестувати такі авто менше. Тому продавці, як правило, просувають їх на таких ресурсах, як OLX та RST, де менше можливостей для реклами. Тому покупці шукатимуть такі авто саме там, не вдаючись до додаткових перевірок.

Спільною рисою всіх ринків є тенденція до масштабного шахрайства щодо пробігу транспортних засобів. Наразі лише AUTO.RIA бореться з цим явищем, пропонуючи своїм клієнтам різні види безкоштовних та додаткових перевірок транспортних засобів.

З порівняльної характеристики (Таблиця 1.1) можна зробити один висновок, ці всі веб – сервіси об'яв з продажу авто дуже схожі за функціоналом і водночас у них є одна спільна проблема, а саме, немає можливості порівняння ціни.

1.4 Розробка архітектури додатку

Веб – сервіс буде реалізований на PHP фреймворку Yii 2.

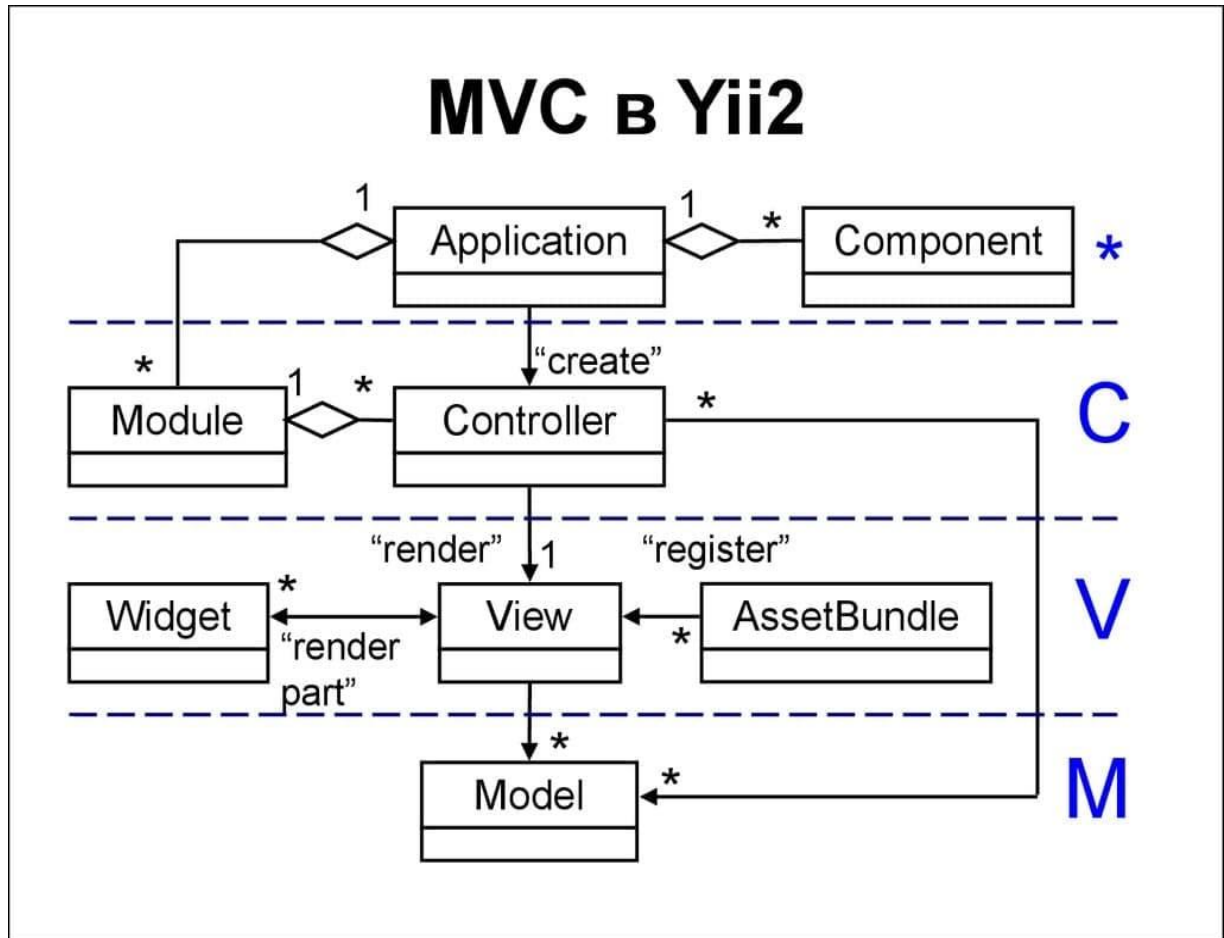


Рис. 1.13 – Архітектура Yii 2

Yii - це швидкий, безпечний та ефективний фреймворк. Він гнучкий, але водночас практичний. Він готовий до використання одразу після отримання. Завдяки розумним інструментам за замовчуванням, Yii забезпечує максимальну функціональність за найнижчу можливу кількість витрат. Розумні значення за замовчуванням і вбудовані інструменти допомагають писати надійний і безпечний код. Він дозволяє писати більше коду за менший час за допомогою простих, але потужних API та генерації коду.

MVC - архітектурний шаблон, що використовується при проектуванні та розробці програмного забезпечення. Архітектура MVC розділяє код програми на три частини. Model, Viewer і Controller, вперше описані в 1978

році, призначалися для програм з графічними інтерфейсами, такими як вікна і кнопки, але пізніше були адаптовані для веб-додатків.

MVC застосовується до різних типів додатків: веб-додатків і десктопних (клієнтських) додатків. Різниця між ними полягає в тому, що у веб-додатку програма отримує один запит від користувача, обробляє його, відображає результат (зазвичай веб-сторінку) і завершує роботу. Якщо надходить інший запит, запускається нова, незалежна копія програми, яка обробляє цей запит. На відміну від веб-додатків, десктопні та мобільні додатки (а також javascript-додатки, що працюють на сторінках браузера) мають тривалий термін служби. Ці додатки обробляють численні запити від користувачів і оновлюють інформацію на екрані без збоїв.

1.5 Обґрунтування вибору технологій

Додаток повинен бути розроблений на PHP.

PHP може робити все; його основне застосування - написання сценаріїв на стороні сервера. Таким чином, PHP може робити те, що роблять інші CGI-програми, наприклад, обробляти дані форм, створювати динамічні сторінки, надсилати та отримувати файли cookie. Однак, PHP може набагато більше.

PHP доступний для багатьох операційних систем, таких як Linux, HP-UX, Solaris, OpenBSD та інших Unix, Microsoft Windows, macOS, RISC OS тощо. PHP також підтримує Apache, IIS та багато інших сучасних веб-серверів. PHP також підтримує Apache, IIS та багато інших сучасних веб-серверів. В принципі, можна використовувати будь-який веб-сервер, який може використовувати двійковий файл PHP FastCGI, наприклад, lighttpd або nginx - PHP може бути запущений як модуль або діяти як CGI-процесор.

MySQL розробляється і підтримується компанією Oracle, яка придбала торгову марку у компанії Sun Microsystems, що придбала шведську компанію MySQL AB. Продукт розповсюджується під ліцензією GNU General Public License та власною комерційною ліцензією. Крім того, розробники іноді створюють функції у відповідь на запити ліцензованих користувачів. Завдяки

цим запитам механізм реплікації з'явився майже у всіх ранніх версіях продукту.

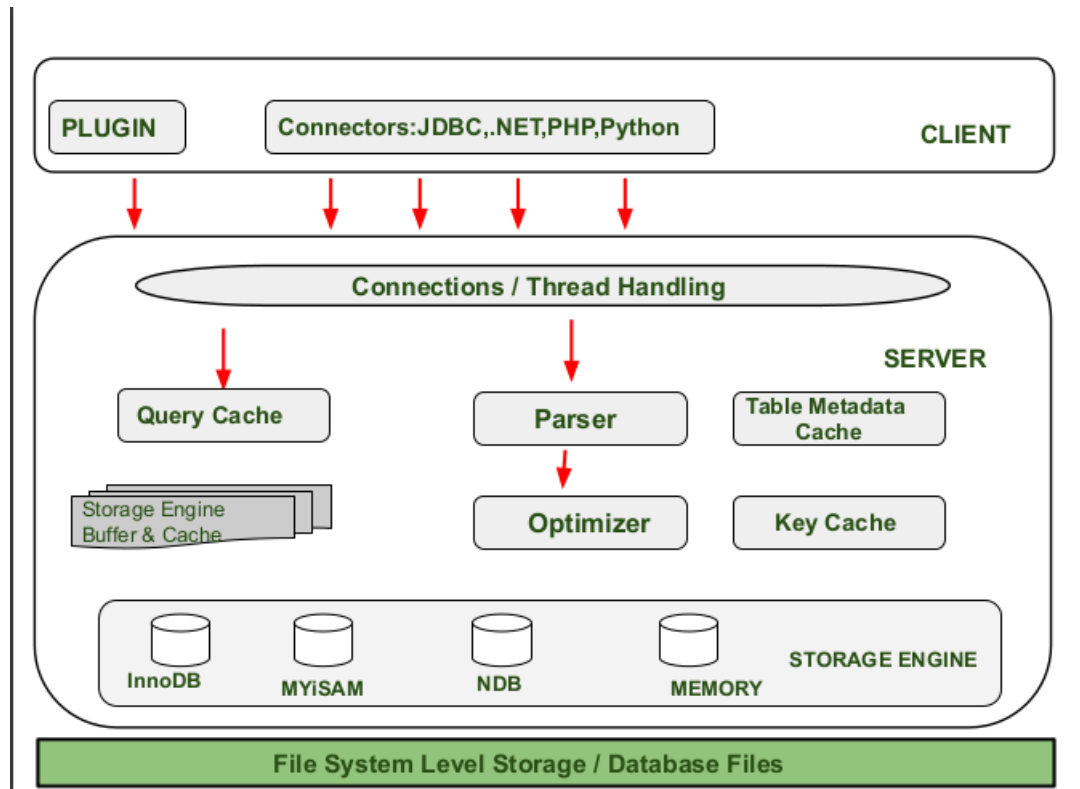


Рис. 1.14 – Архітектура MySQL

Гнучкість MySQL забезпечується підтримкою великої кількості типів таблиць. Користувачі можуть вибирати між таблицями MyISAM, які підтримують повнотекстовий пошук, і таблицями InnoDB, які підтримують транзакції лише на рівні окремих записів. Крім того, в MySQL є спеціальний тип таблиць EXAMPLE, який надає принципи створення нових типів таблиць. Завдяки відкритій архітектурі та ліцензії GPL до MySQL постійно додаються нові типи таблиць.

SQL (Structured Query Language - мова структурованих запитів) - це мова програмування для зберігання та обробки інформації в реляційних базах даних. Реляційні бази даних зберігають інформацію у вигляді таблиць, рядки і стовпці яких представляють різні атрибути даних і різні зв'язки між значеннями даних. Команди SQL можна використовувати для зберігання, оновлення, видалення, пошуку та отримання інформації в базі даних. SQL

також можна використовувати для підтримки та оптимізації продуктивності бази даних.

Мова структурованих запитів (SQL) - це поширена мова запитів, яка використовується у всіх типах додатків. Аналітики даних і розробники вивчають і використовують SQL, оскільки вона добре інтегрується з різними мовами програмування. Наприклад, ви можете реалізувати SQL-запити на мові програмування Java і створювати високопродуктивні додатки для обробки даних на великих системах баз даних SQL, таких як Oracle і MS SQL Server. SQL-рішення дуже легко вивчити, оскільки в операторах використовуються загальноприйняті англійські ключові слова SQL-рішення дуже легко вивчити, оскільки вони засновані на загальноприйнятих англійських ключових словах.

JavaScript - це мова програмування, яка додає інтерактивності веб-сайтам (ігри, зворотній зв'язок від натискання кнопок і введення даних у форми, динамічні стилі, анімацію тощо). Ця стаття допоможе вам почати роботу з цією захоплюючою мовою і дізнатися, що вона може зробити для вас.

JavaScript напрочуд гнучка та зручна для початківців. З вашим досвідом ви зможете створювати ігри, 2D та 3D анімацію, інтегровані додатки для роботи з базами даних та багато іншого.

Сам по собі JavaScript дуже лаконічний, але надзвичайно гнучкий. Розробники написали низку інструментів на основі основної мови JavaScript, які розкривають велику кількість додаткових функцій без особливих зусиль.

HTML (мова розмітки гіпертексту) - це код, який використовується для структурування і відображення веб-сторінок та їхнього вмісту. Наприклад, вміст можна структурувати за допомогою абзаців, маркованих списків, зображень і таблиць даних. Як випливає з назви, ця стаття надає базове розуміння HTML та її функцій.

HTML - це не мова програмування, а мова розмітки, яка використовується для того, щоб вказати браузеру, як відображати веб-

сторінку, до якої ви звертаєтесь. HTML може бути настільки складним або простим, наскільки хоче дизайнер веб-сайту; він складається з ряду елементів, які використовуються для структурування або обгортання різних фрагментів контенту таким чином, щоб вони виглядали і поведилися певним чином. За допомогою тегів безпеки можна зробити так, щоб слово або зображення посилалося на щось інше, виділити слово курсивом, зробити шрифт більшим або меншим тощо.

Як і HTML, CSS - це не мова програмування. CSS - це не мова розмітки, а скоріше мова таблиць стилів. Це означає, що стилі можна вибірково застосовувати до елементів у документі HTML.

CSS найчастіше використовується для візуального представлення сторінок, написаних у форматах HTML і XHTML, але може застосовуватися і для інших типів XML-документів.

Висновок до першого розділу

У даному розділі були розглянуті основні поняття про автомобільний ринок України, проведено детальний аналіз та дослідження. Також була проведена порівняльна характеристика всіх існуючих україномовних аналогів в ході якої було виявлено що всі аналоги схожі між собою але лідером по технологіям являється «Auto.ria».

РОЗДІЛ 2. МЕТОДИ ТА АРХІТЕКТУРА ВЕБ – СЕРВІСУ ОБ’ЯВ З ПРОДАЖУ АВТО З МОЖЛИВІСТЮ ПОРІВНЯННЯ ЦІНИ

2.1 Методи та аналіз сервісу порівняння цін

В мережі Інтернет сервіси порівняння цін дозволяють користувачам бачити список різних цін на певний товар. Більшість сервісів порівняння цін самі не продають товар, а отримують ціни від роздрібних торговців, у яких користувачі можуть його придбати. У Великій Британії, наприклад, у 2005 році ці послуги принесли дохід до 130 мільйонів фунтів стерлінгів.

Зазвичай такі сервіси не мають власної продукції та мають прибуток виключно від реклами на власному порталі чи від інших сервісів які співпрацюють із сервісом порівняння цін. Тобто звичайні онлайн магазини зв’язуються з сервісом порівняння цін, для того щоб на цьому сервісі була відображена продукція магазину. Після успішного обговорення деталей магазин ділиться API – ключем.

API - це механізм, який дозволяє двом програмним компонентам взаємодіяти один з одним за допомогою набору визначень і протоколів. Наприклад, метеорологічна система зберігає щоденні дані про погоду. Додаток для прогнозування погоди на смартфоні зв'язується з цією системою через API і відображає щоденний прогноз погоди на смартфоні.

У контексті API термін "додаток" означає будь-яке програмне забезпечення з певною функціональністю. Інтерфейс можна уявити як договір про надання послуг між двома додатками. Цей контракт визначає, як вони взаємодіють один з одним за допомогою запитів і відповідей; документація API надає інформацію про те, як розробники повинні структурувати ці запити і відповіді.

Спочатку розглянемо, як себе веде користувач, який не використовує сервіс порівняння ціни.

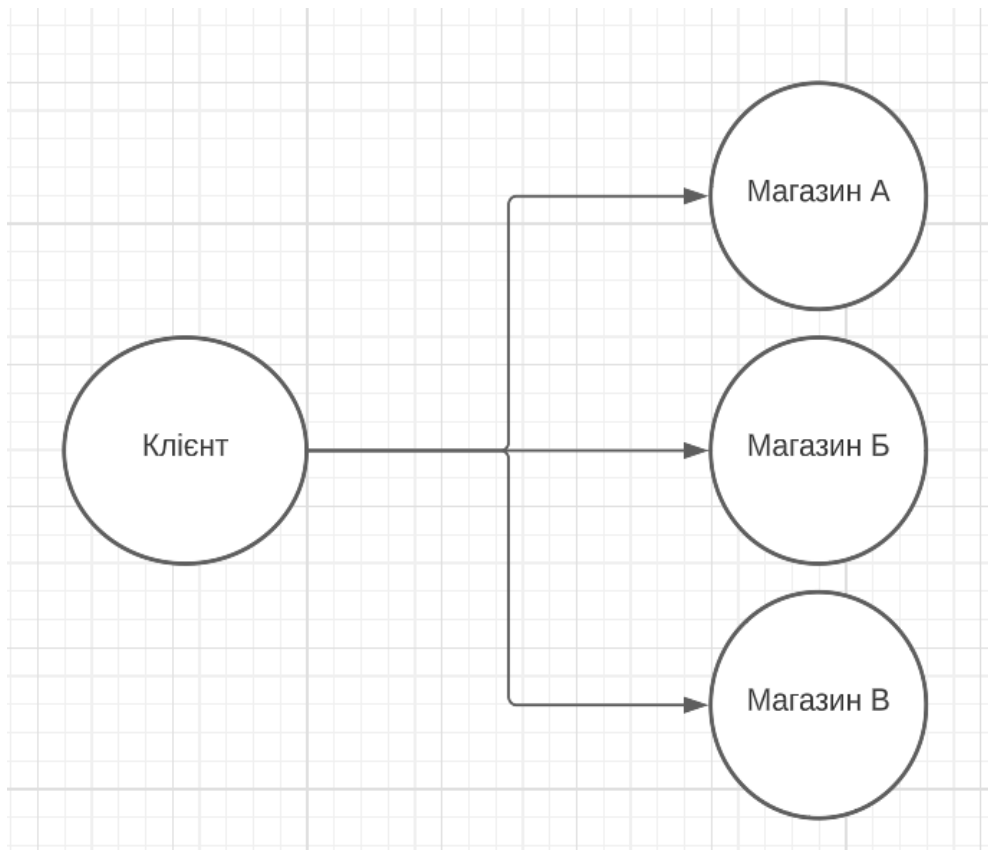


Рис. 2.1 – Схема можливостей користувача

Як ми можемо бачити, користувачу, який не використовує сервіс порівняння цін, потрібно спочатку зайти на веб сторінку магазину А, шукати потрібний товар, запам'ятати ціну, зайти на веб сторінку магазину Б і повторити попередні дії і т.д., а потім уже вибирати товар враховуючи ціну, на всіх сервісах. Для користувача це дуже незручно, так як на кожному з цих магазинів потрібно буде вписувати в пошук одні і ті ж самі слова, щоб знайти що його цікавить. Тому дана стратегія не є оптимальною для нього.

На наступній схемі(Рис. 2.2) розглянемо можливості користувача який піде іншим шляхом.

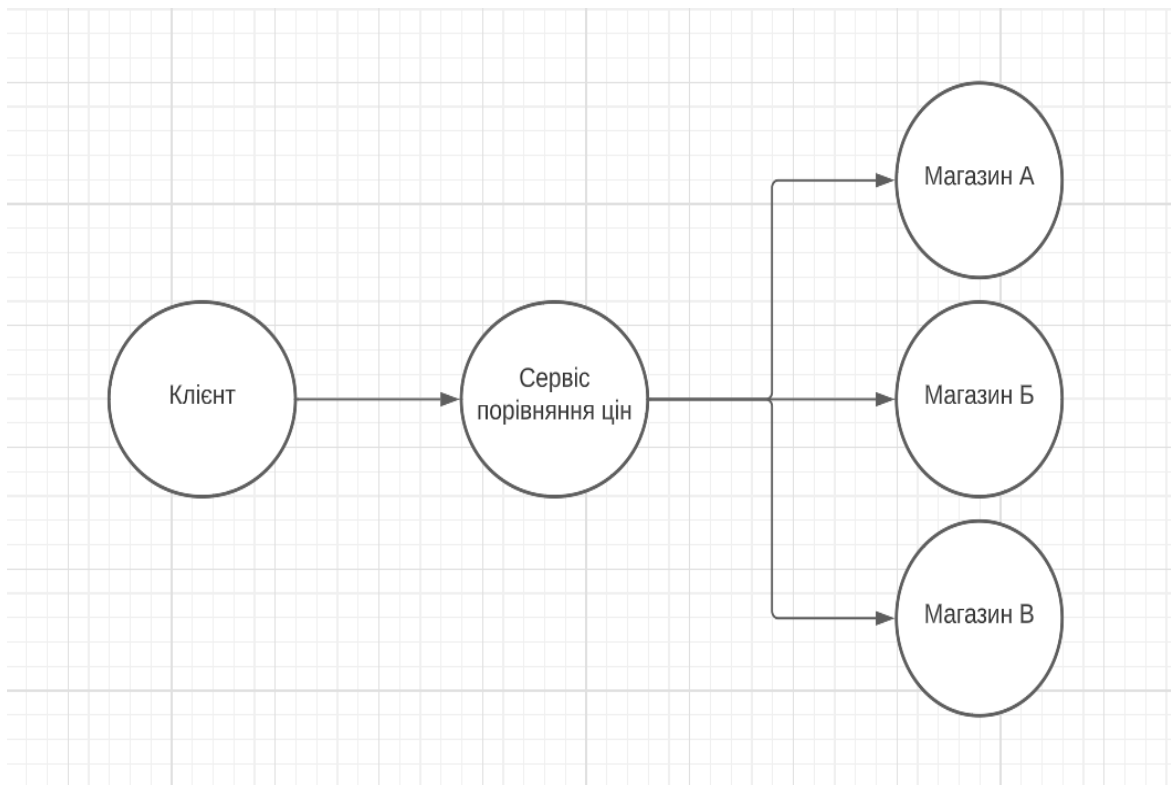


Рис. 2.2 – Схема можливостей користувача з використанням сервісу порівняння цін

Дана схема можливостей є клієнт орієнтованою, тому що користувачу більше не потрібно заходити на веб сторінки всіх магазинів, достатньо зайти на сайт цього сервісу і клієнт, один раз вписавши в пошук потрібні дані, побачить усі оголошення, які його цікавлять.

На сторінці того ж самого веб – сервісу порівняння цін, користувач зможе побачити і порівняти усі дані, фото, стан автомобілів і якщо він визначиться з вибором то тільки тоді він перейде на веб сторінку того магазину і далі зв’яжеться з власником авто.

Для того щоб магазин знав що користувач перейшов на їх веб сайт з нашого веб – сервісу оголошень з продажу авто з можливістю порівняння цін, потрібно зробити так щоб сервер магазину «зрозумів» що користувач перейшов саме з нашого сервісу. Це можна зробити за допомогою методу GET або POST. Детальніше розглянемо обидва методи за допомогою визначень та діаграми порівняння (Таблиця 2), для того щоб більш детально ознайомитись з двома методами.

Таблиця 2.1

Основа для порівняння	GET	POST
Параметри розміщені всередині	URI	Тіло
Призначення	Отримання документів	Оновлення даних
Результати запиту	Здатний бути закладкою.	Неможливо зробити закладку.
Безпека	Вразливий, як присутній у відкритому тексті	Безпечніше, ніж метод GET
Обмеження типу даних форми	Дозволяються лише символи ASCII.	Не допускається жодних обмежень, навіть двійкових даних.
Довжина даних форми	Має бути максимально мінімальним.	Може лежати в будь-якому діапазоні.
Видимість	Може бачити кожен.	Не відображаються змінні в URL-адресі.
Змінна розмір	До 2000 символів.	До 8 Мб
Кешування	Дані методу можна кешувати.	Не кешує дані.

GET-запит - це метод передачі даних від клієнта до сервера для отримання інформації, заданої певними GET-параметрами. Це загальнодоступні дані, які можна використовувати для відображення посилань історії. Цей запит підходить для використання, якщо дані в адресному рядку не змінювалися. Щоб додати GET-параметри до запиту, додайте "?" до URL-адреси і почніть конфігурацію за наступними правилами.

ім'я_параметра1=значення_параметра1&ім'я_параметра2=значення_параметра2&... Розділювачем між параметрами є символ "&".

Метод POST-запитів використовується для надсилання запиту, який веб - сервер приймає для зберігання даних, що містяться в тілі повідомлення. Цей метод часто використовується для завантаження файлів або відправки заповнених веб - форм. На відміну від нього, метод HTTP GET призначений для отримання інформації з сервера. Щоб виконати цей тип запиту, ви повинні натиснути на кнопку з атрибутом type = "submit" на веб-сторінці. Зверніть увагу, що ця кнопка знаходиться всередині елемента <form>, а атрибут method має значення post.

2.2 Моделювання процесів веб – додатку

Згідно до вимог можна побудувати діаграму варіантів використання.

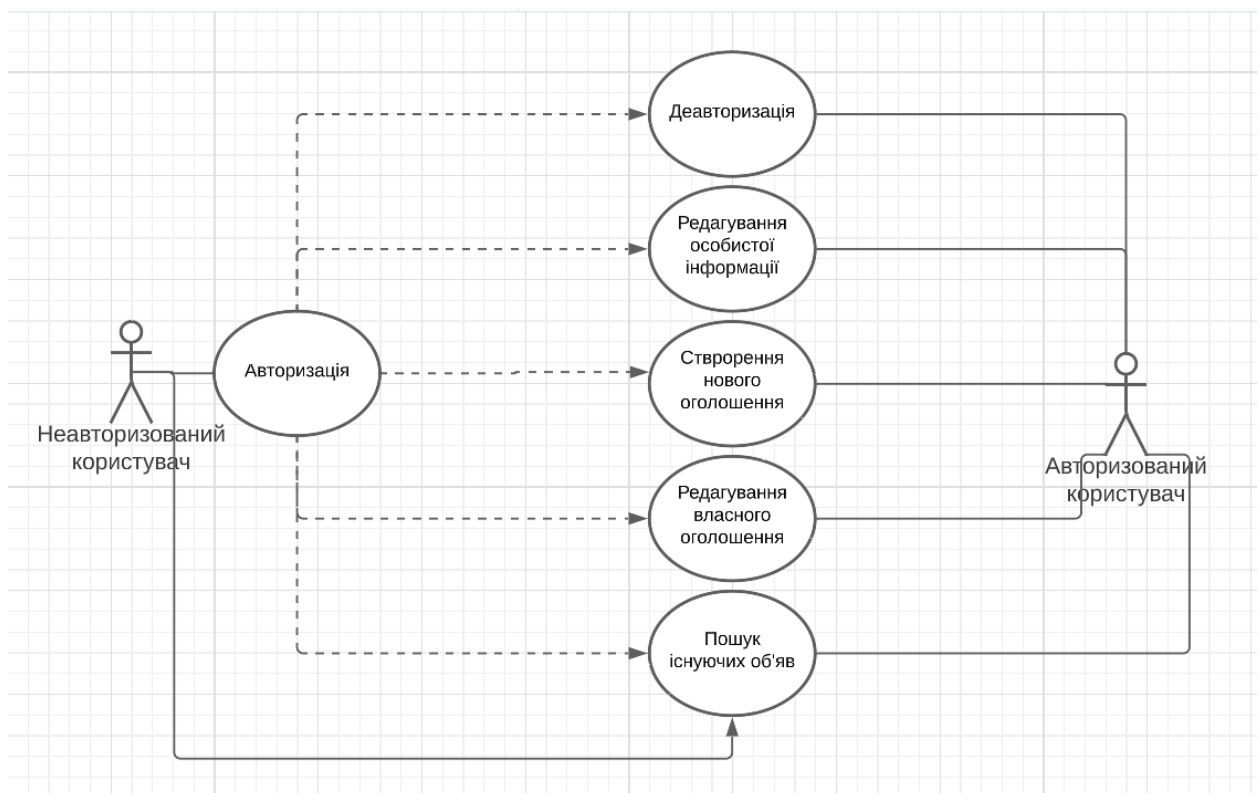


Рис. 2.3 – Діаграма варіантів використання

Неавторизований користувач:

- Авторизація
- Пошук існуючих об'єктів

Авторизований користувач:

- Деавторизація
- Редагування особистої інформації
- Створення нового оголошення
- Редагування власного оголошення
- Пошук існуючих об'яв

У веб – сервісі об'яв з продажу авто з можливістю порівняння ціни, користувач має можливість переглядати оголошення з різних авторинків вживаних авто і одразу ж побачити ціни на них. Після авторизації клієнт має можливість створити власне оголошення та після цього відредагувати його за потребою.

2.3 Розробка бази даних для веб – додатку

В якості бази даних використовується MySQL. Ця база даних найчастіше використовується для зберігання даних на веб - сторінках, які створюються. Під час розробки бази даних було створено 13 таблиць. Призначення і найменування кожної із таблиць описано в таблиці 2.2.

Інформація про таблиці бази даних.

Таблиця 2.2

Назва таблиці	Призначення
bodyType	Таблиця містить типи кузовів авто
carImages	Таблиця містить шлях до зображень
carInfo	Таблиця містить всю інформацію про оголошення
carModel	Таблиця містить назви моделей авто
carState	Таблиця містить назви станів авто
carWheelDrive	Таблиця містить назви типів приводів авто
engineType	Таблиця містить назви типів двигунів
fuelType	Таблиця містить назви типів пального

migration	Таблиця містить дані про маграції таблиць
Regions	Таблиця містить назви областей України
tradeMark	Таблиця містить назви торгівельних марок
transmissonType	Таблиця містить назви типів коробок передач
user	Таблиця містить всю інформацію про зареєстрованих користувачів

На рисунку 2.3 представлена діаграма бази даних

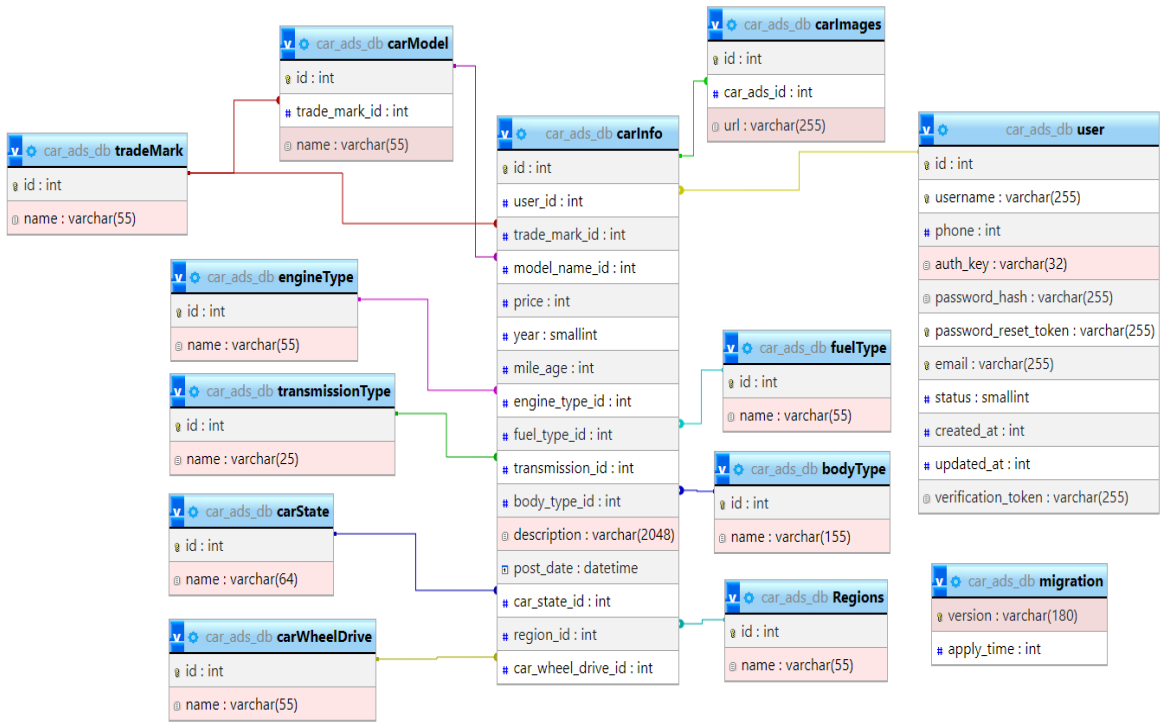


Рис. 2.4 – Діаграма бази даних

Нижче наведено опис основних таблиць бази даних.(Таблиці 2.3 – 2.16)

Таблиця 2.3

Опис полів таблиці «bodyType»

Назва	Тип даних	PK	FK	Опис поля

Продовження таблиці 2.3

id	int	+	-	Ідентифікатор типу кузова
name	varchar	-	-	Найменування типів кузовів авто

Таблиця bodyType відповідає за збереження інформації про типи кузовів для автомобілів.

Таблиця 2.4

Опис полів таблиці «carImages»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор зображення
car_ads_id	int	-	+	Найменування типів кузовів авто
url	varchar	-	-	Шлях до фото

Таблиця carImages відповідає за збереження шляху до зображення авто, та зберігання інформації до якої об'яви належить це зображення.

Таблиця 2.5

Опис полів таблиці «carInfo»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор оголошення
user_id	int	-	+	Ідентифікатор власника оголошення
trade_mark_id	int	-	+	Ідентифікатор торгівельної марки

Продовження таблиці 2.5

model_name_id	int	-	+	Ідентифікатор назви моделі
price	int	-	-	Ціна
year	smallint	-	-	Рік виготовлення авто
mile_age	int	-	-	Пробіг авто
engine_type_id	int	-	+	Ідентифікатор типу двигуна
fuel_type_id	int	-	+	Ідентифікатор типу палива
transmission_id	int	-	+	Ідентифікатор типу коробки передач
body_type_id	int	-	+	Ідентифікатор типу кузова
description	varchar	-	-	Опис до оголошення
post_date	datetime	-	-	Дата створення оголошення
region_id	int	-	+	Ідентифікатор області України
car_wheel_drive_id	int	-	+	Ідентифікатор типу приводу авто

Таблиця carInfo відповідає за збереження інформації про оголошення з продажу авто.

Таблиця 2.6

Опис полів таблиці «carModel»

Назва	Тип даних	PK	FK	Опис поля

Продовження таблиці 2.6

id	int	+	-	Ідентифікатор моделі
trade_mark_id	int	-	+	Ідентифікатор торгівельної марки
name	varchar	-	-	Найменування моделі

Таблиця carModel відповідає за збереження інформації про моделі авто згідно торгівельної марки автомобіля.

Таблиця 2.7

Опис полів таблиці «carState»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор стану авто
name	varchar	-	-	Найменування стану

Таблиця carState відповідає за збереження інформації про види станів автомобіля.

Таблиця 2.8

Опис полів таблиці «carWheelDrive»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор приводу авто
name	varchar	-	-	Найменування приводу авто

Таблиця carWheelDrive відповідає за збереження інформації про види приводів у авто.

Таблиця 2.9

Опис полів таблиці «engineType»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор типу двигуна

Продовження таблиці 2.9

name	varchar	-	-	Значення об'єму двигуна
------	---------	---	---	-------------------------

Таблиця engineType відповідає за збереження інформації про типи двигунів, а саме, їх об'єм.

Таблиця 2.10

Опис полів таблиці «fuelType»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор типу палива
name	varchar	-	-	Найменування палива

Таблиця bodyType відповідає за збереження інформації про види палив для авто.

Таблиця 2.11

Опис полів таблиці «migrations»

Назва	Тип даних	PK	FK	Опис поля
version	varchar	+	-	Версія міграції
apply_time	int	-	-	Час застосування

Таблиця migration відповідає за збереження інформації про всі здійснені маграції таблиць.

Таблиця 2.12

Опис полів таблиці «regions»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор області України
name	varchar	-	-	Найменування області України

Таблиця regions відповідає за збереження найменувань всіх областей України.

Таблиця 2.13

Опис полів таблиці «tradeMark»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор торгівельної марки
name	varchar	-	-	Найменування торгівельної марки

Таблиця tradeMark відповідає за збереження найменувань торгівельних марок авто.

Таблиця 2.14

Опис полів таблиці «transmissonType»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор типу коробки передач
name	varchar	-	-	Найменування типу коробки передач

Таблиця transmissonType відповідає за збереження найменувань типів коробок передач для авто.

Таблиця 2.15

Опис полів таблиці «user»

Назва	Тип даних	PK	FK	Опис поля
id	int	+	-	Ідентифікатор користувача
username	varchar	-	-	Найменування користувача
phone	int	-	-	Номер телефону
auth_key	varchar	-	-	Ключ авторизації
password_hash	varchar	-	-	Хеш паролю

Продовження таблиці 2.16

password_reset_token	varchar	-	-	Токен для зміни паролю
email	varchar	-	-	Електронна адреса
status	smallint	-	-	Статус користувача
created_at	int	-	-	Дата реєстрації
updated_at	int	-	-	Дата останнього оновлення даних
verification_token	varchar	-	-	Токен верифікації користувача

Таблиця user відповідає за збереження ключової інформації про користувачів.

Перелік об'єктів бази даних, до яких можуть мати доступ користувачі бази даних, наведено в таблиці 2.19. На перетині рядків і стовпців вказані дії, які може виконувати даний тип користувача: 0 - Немає доступу, 1 - Перегляд даних, 2 - Редагування, 3 - Видалення, 4 - Вставка даних, 5 - Повний доступ (виконання процедури).

Таблиця 2.17

Таблиця доступу до бази даних

Таблиця	Гість	Користувач	Адміністратор
bodyType	1	1	1,2,3,4,5
carImages	1	1	1,2,3,4,5
carInfo	1	1,2,3,4	1,2,3,4,5
carModel	1	1	1,2,3,4,5
carState	1	1	1,2,3,4,5
carWheelDrive	1	1	1,2,3,4,5
engineType	1	1	1,2,3,4,5
fuelType	1	1	1,2,3,4,5
migrations	0	0	1,2,3,4,5
regions	1	1	1,2,3,4,5

Продовження таблиці 2.17

tradeMark	1	1	1,2,3,4,5
transmissionType	1	1	1,2,3,4,5
user	0	1,2,3,4	1,2,3,4,5

Таким чином, база даних реалізована відповідно до вимог стабільності та безперервного доступу до даних. У базі даних реалізовано 13 таблиць, з яких основними є 2, а саме: carinfo, user.

2.4 Проектування та реалізація алгоритмів веб – додатку

Фреймворк Yii 2 має шаблон проектування MVC (рис. 2.5), який дозволяє швидко і легко розробляти додатки.

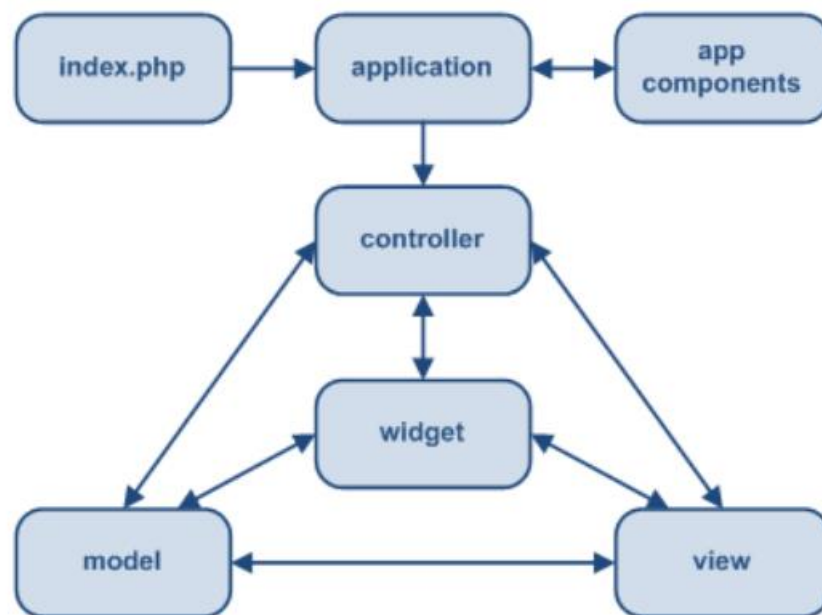


Рис. 2.5 – Операційна структура роботи фреймворку Yii 2

Як було сказано вище, для реалізації проекту використовується php.Yii Framework 2 (advanced). Ця версія фреймворку має три різних інтерфейси:

- Frontend – використовується для взаємодії з користувачем
- Backend – використовується для взаємодії з адміністратором
- Console – взаємодія з користувачем через командний рядок

Для нашого веб – додатку було реалізовано діаграма класів моделей(class diagram) (рис.2.6).

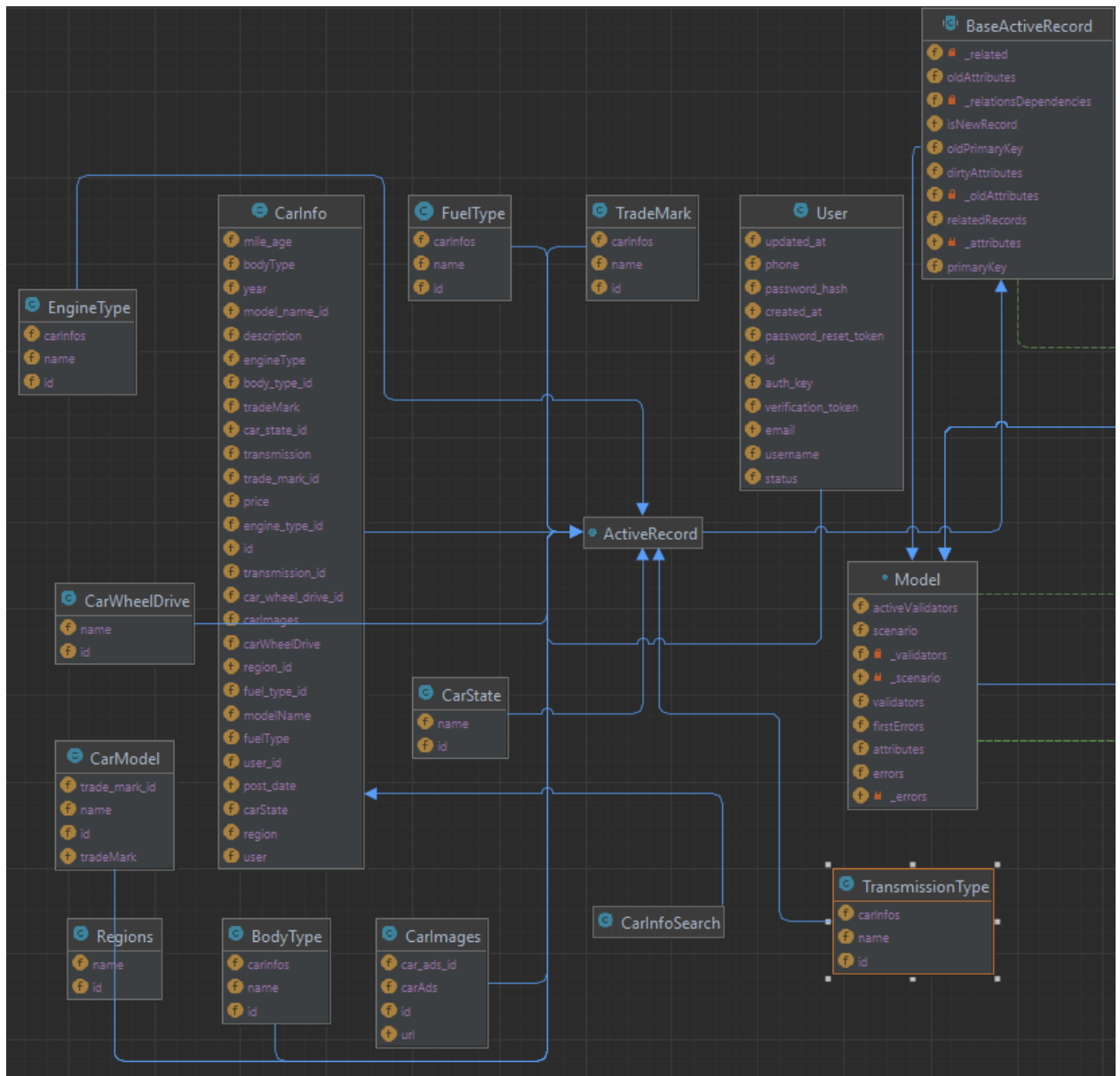


Рис. 2.6 – Діаграма класів моделей

Усі розроблені класи об'єктної моделі успадковують клас Active Record. Цей клас є об'єктом шаблону проектування, який використовується для реалізації доступу до реляційних баз даних. Завдяки цьому класу можна швидко взаємодіяти з базою даних сайту. Розглянемо докладніше класи, описані вище:

- BodyType – клас об'єкта типів кузовів;
- CarImages – клас об'єкта зображень оголошень;

- CarInfo – клас об'єкта інформації про об'яву;
- CarInfoSearch – клас об'єкта пошуку оголошень;
- CarModel – клас об'єкта типів моделей авто;
- CarState – клас об'єкта стану авто;
- CarWheelDrive – клас об'єкта типу приводу авто;
- EngineType – клас об'єкта типів двигунів;
- FuelType – клас об'єкта типу пального;
- PasswordResetRequestForm – клас об'єкта зміни паролю;
- Regions – клас об'єкта області України;
- ResendVerificationEmailForm – клас об'єкта повторної відправки листа для підтвердження реєстрацію;
- SignupForm – клас об'єкта форми реєстрації;
- TradeMark – клас об'єкта найменування торгівельної марки;
- TransmissionType – клас об'єкта типу коробки передач;
- User – клас об'єкта користувача;
- VerifyEmailForm – клас об'єкта відправки повідомлення на пошту для підтвердження реєстрації;

Для взаємодії між інтерфейсом та моделлю створюються контролери. Для реалізації частини для зовнішніх користувачів були розроблені наступні класи контролерів, як показано на рисунку 2.7.

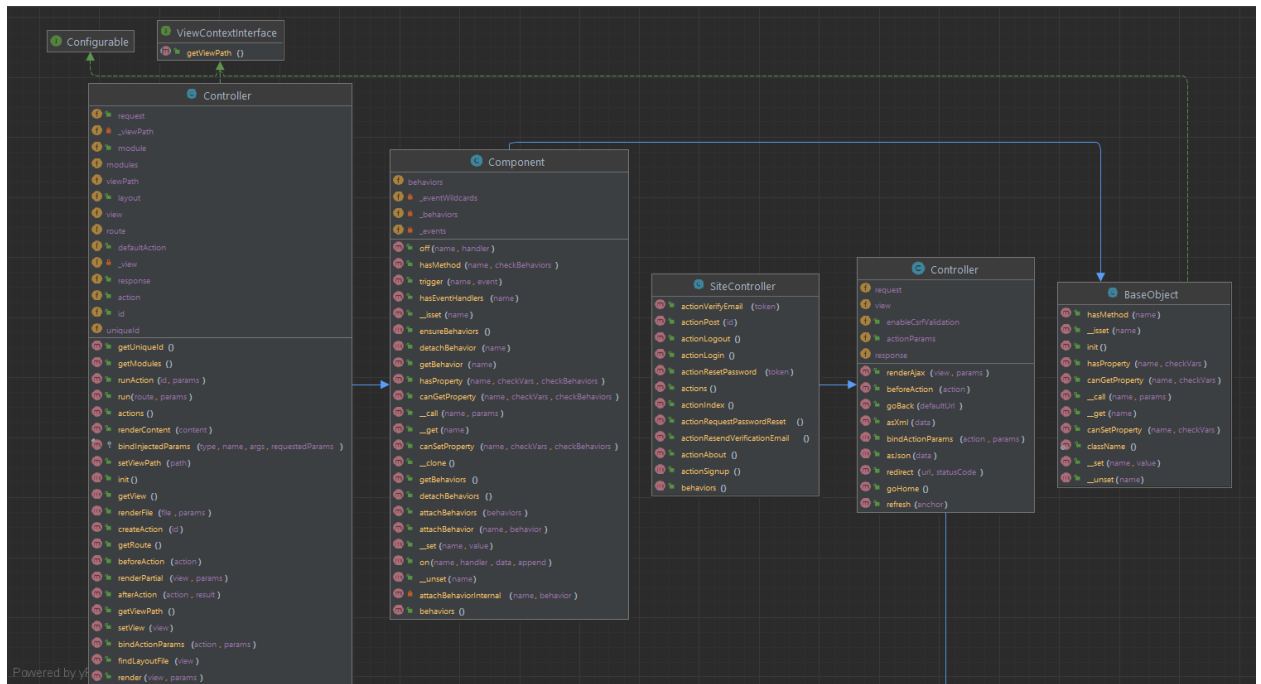


Рис. 2.7 – Діаграма класів контролерів

На даній діаграмі представлено клас SiteController.

Контролер є частиною архітектури MVC. Контролер - це об'єкт класу, який успадковується від yiiBaseController і відповідає за обробку запитів та генерування відповідей. Зокрема, після отримання контролю над додатком, контролер аналізує дані вхідного запиту, передає їх моделі, вставляє результати роботи моделі у вьюшку і, нарешті, генерує вихідну відповідь.

Контролер складається з дій, які є основними блоками, до яких кінцевий користувач може отримати доступ і запросити виконання певних операцій. Контролер може мати одну або декілька дій.

SiteController – це контролер, який відповідає за обробку запитів та генерацію відповідей.

Розглянемо його детальніше. SiteController має:

- actionVerifyEmail – метод, який відповідає за відправку листа на електронну адресу для підтвердження реєстрації;
- actionPost – метод, який відповідає за відображення конкретного оголошення;
- actionLogout – метод, який відповідає за деавторизацію користувача;

- `actionLogin` – метод, який відповідає за авторизацію користувача на сайті;
- `actionResetPassword` – метод, який відповідає за зміну пароллю користувача;
- `actionIndex` – метод, який відповідає за відображення головної сторінки;
- `actionRequestPasswordReset` – метод, який відповідає за обробку запиту на зміну пароллю користувача;
- `actionResentVerificationEmail` – метод, який відповідає за повторну відправку листа на електронну адресу користувача для підтвердження реєстрації;
- `actionAbout` – метод, який відповідає за відображення сторінки «Про нас»
- `actionSignup` – метод, який відповідає за реєстрації користувача на сайті;

Розглянемо алгоритми роботи та стани програмної системи за допомогою діаграми діяльності (activity diagram).

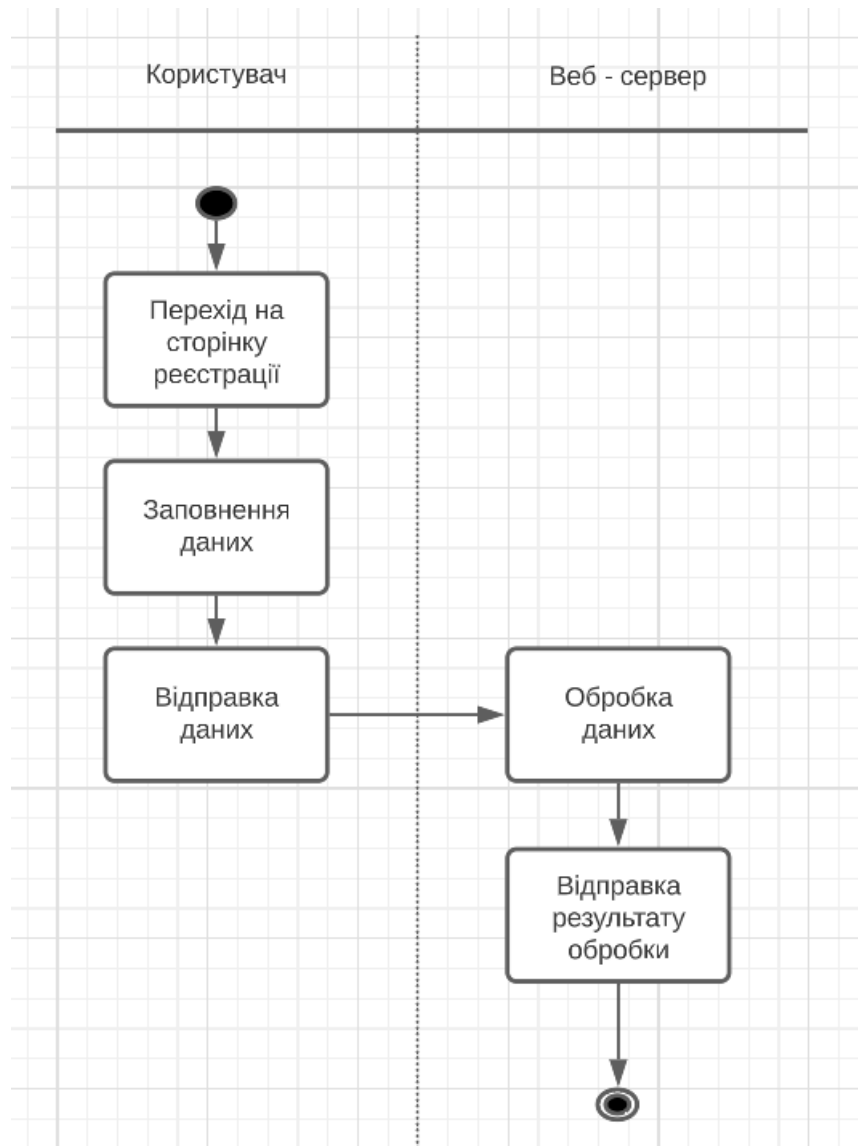


Рис. 2.8 – Діаграма створення профілю

На рисунку 2.8 представлена діаграма діяльності для створення профілю.

Описання діаграми активності (Створення профілю):

1. Перехід на сторінку реєстрації
2. Заповнення даних
3. Відправка даних
4. Обробка даних
5. Відправка результату обробки

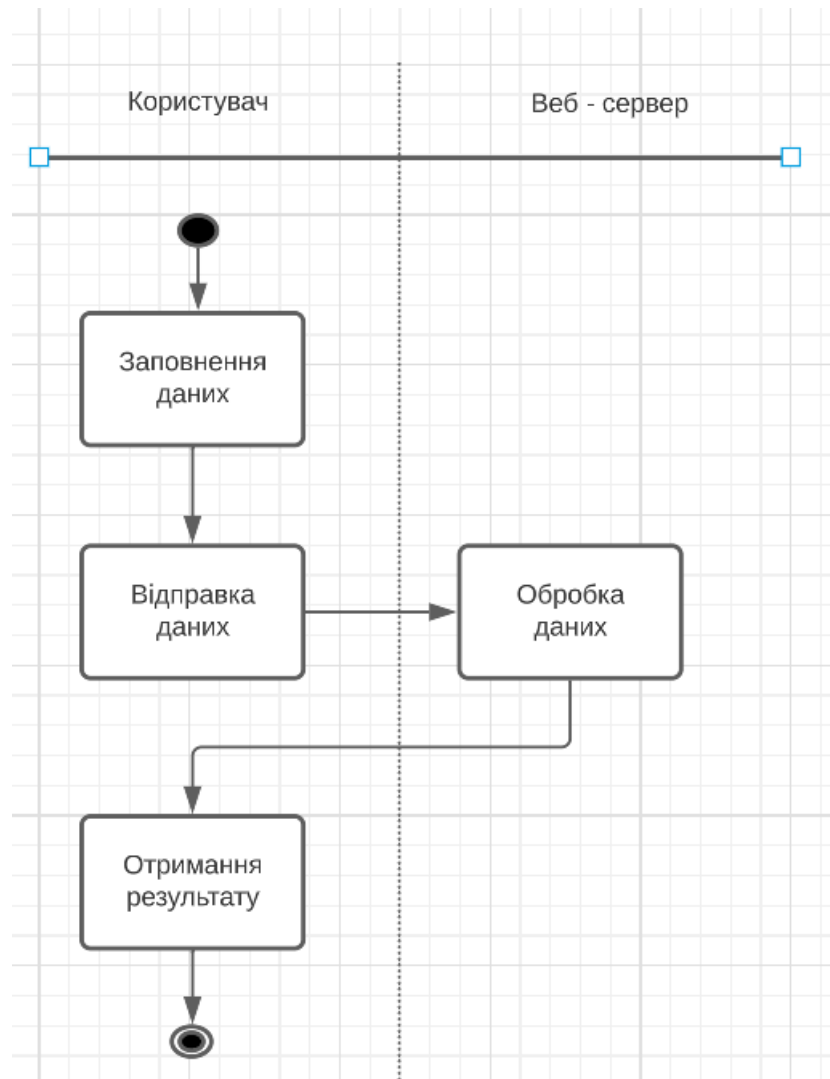


Рис. 2.9 – Діаграма редагування опису оголошення

Описання діаграми активності (редагування опису оголошення):

1. Заповнення даних
2. Відправка даних
3. Обробка даних
4. Отримання результату

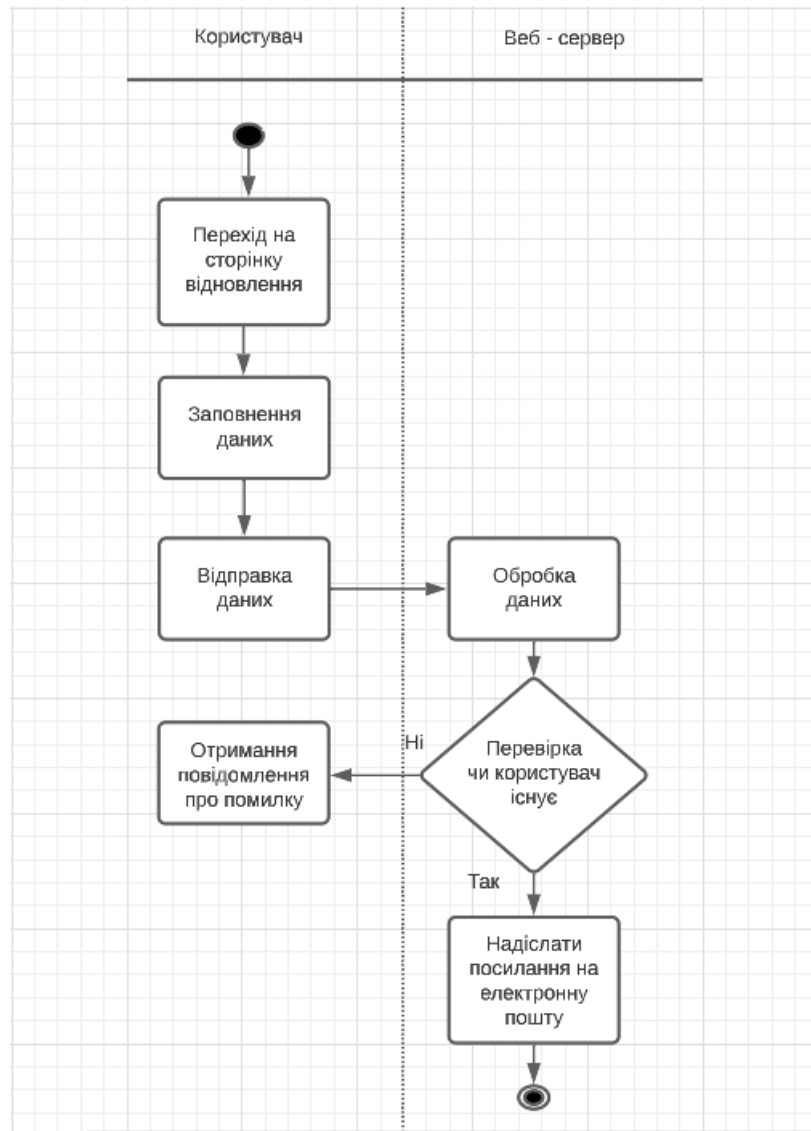


Рис. 2.10 – Діаграма відновлення паролю

Описання діаграми активності (редагування опису оголошення):

1. Перехід на сторінку відновлення
2. Заповнення даних
3. Відправка даних
4. Обробка даних
5. Надсилання листа з посиланням на електронну адресу

2.5 Проектування та реалізація алгоритмів веб – додатку

Для підключення до бази даних в Yii 2 проєкті потрібно в файлі `main-local.php` внести зміни, а саме:

1. Назву хосту (В нашому випадку localhost)
2. Назву бази даних

3. Ім'я користувача
4. Пароль для входу
5. Кодування

```
'db' => [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=car_ads_db',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
],
```

В тому ж самому файлі налаштуємо swiftmailer щоб листи відправлялись з gmail пошти, для цього потрібно внести наступні зміни в файлі:

1. Вказати клас Swift_SmtpTransport
2. Вказати хост smtp.gmail.com
3. Вказати пошту з якої будуть надсилатися повідомлення
4. Вказати пароль до пошти
5. Вказати порт 587
6. Вказати кодування tls

Таблиці описуються в папці Models, в php класах, які наслідують клас ActiveRecord. Кожен клас має мати одну і ту ж саму назву з таблицею. Найкращим шляхом буде, згенерувати кожну модель за допомогою інструменту Model Generator. Опис полів таблиці має наступний вигляд:

```
return [
    [['username', 'phone', 'auth_key', 'password_hash', 'email',
'created_at', 'updated_at'], 'required'],
    [['phone', 'status', 'created_at', 'updated_at'], 'integer'],
    [['username', 'password_hash', 'password_reset_token', 'email',
'verification_token'], 'string', 'max' => 255],
    [['auth_key'], 'string', 'max' => 32],
    [['username'], 'unique'],
    [['email'], 'unique'],
    [['password_reset_token'], 'unique'],
];
```

Для створення route в Yii 2, потрібно в контролері написати метод, який називається action + назву сторінки без пробілів. Наприклад: actionAboutUs(). Для відображення представлення (view), яке має називатися

так само як наш route, його потрібно вказати в лапках в методі render(). Якщо в назва route має два слова, тоді при виклику методу render потрібно вказати ці два слова через тире.

```
public function actionAboutUs()
{
    return $this->render('about-us');
}
```

Для реалізації фільтра пошуку об'яв потрібно використати Ajax запити, для того щоб сторінка не перезавантажувалася при натисканні кнопки пошуку. Відправка Ajax запиту з клієнтської частини має наступний вигляд:

```
$.ajax({
    url: '/frontend/web/index.php?r=site/index',
    type: 'POST',
    data: {
        *Дані для відправки на сервер*
    },
    success: function (res) {
        *Виконання інструкцій у разі успішного запиту*
    },
    error: function (e) {
        *Виконання інструкцій у разі неуспішного запиту*
    }
});
```

Для того щоб обробити отриманні данні на сервері, потрібно у тому роуті на який надсилали Ajax запит, потрібно використати метод isPost(), який повертає true або false. В кінці методу потрібно за допомогою return, надсилаємо відповідь на клієнтську частину. Він має наступний вигляд:

```
if (Yii::$app->request->isPost) {
    if (isset($_POST['sval'])) {
        $data = $_POST['sval'];
        $tradeMark = TradeMark::find()
            ->where(['name' => $data])
            ->one();
        $carModels = CarModel::find()
            ->where(['trade_mark_id' => $tradeMark['id']])
            ->all();
        $carModelsArr = [];
        foreach ($carModels as $carModel) {
            array_push($carModelsArr, $carModel['name']);
        }
        return json_encode($carModelsArr);
    }
}
```

Щоб отримати доступ до пошуку о'б'яв на AutoRia, потрібно скористатися RIA API. Для цього необхідно виконати GET-запит наступного вигляду:

https://developers.ria.com/auto/search?api_key=YOUR_API_KEY&PARAMETERS

Запит складається з:

- SEARCH – назва методу, до якого ми будемо звертатися
- API_KEY – ключ доступу. Щоб отримати ключ доступу, потрібно зареєструватися на платформі developers.ria.com.
- PARAMETERS – вхідним параметром є послідовність пар ім'я=значення, розділених амперсандом. Список параметрів вказаний на сайті для розробників developers.ria.com

Якщо введені вами дані вірні, від серверу прийду відповідь у форматі JSON. Відповідь має наступний вигляд:

```
[
  {
    "additional_params": {
      "lang_id": 2,           // Російська мова
      "page": 0,             // Порядковий номер сторінки
      "view_type_id": 0,
      "target": "search",
      "section": "auto",     // Пошук по авто
      "catalog_name": "",
      "elastica": true,
      "nodejs": true
    },
    "result": {              // Результат пошуку
      "search_result": {
        "ids": [
          "19519211",
          "19684763",
          "19493489",
          "19714833",
          "19393702",
          "19692238",         // id оголошень
          "19574398",
          "18154136",
          "19391327",
          "18693600",
          "19431563",
          "18047892"
        ],
        "count": 1,          // Кількість id оголошень доступних за заданими парам
        "last_id": 0,
        "qs": {
          "fields": [
            "_id"
          ],
          "size": 50,        // Кількість відображуваних id оголошень
          "from": 0,
        }
      }
    }
  }
]
```

Рис. 2.11 – Приклад успішної відповіді серверу

Для отримання інформації про окреме оголошення потрібно знову надіслати GET запит наступного вигляду:

https://developers.ria.com/auto/used/autos/advertisementId?user_id=ВашID&api_key=YOUR_API_KEY

В даному запиті є:

- advertisementId – ідентифікатор оголошення
- user_id – ваш ідентифікатор в системі RIA
- YOUR_API_KEY – ваш ключ доступу

Приклад успішної відповіді:

```

{
  "_id": 20445742,
  "fuel": {
    "consumption": {
      "city": 10,
      "route": 6,
      "combine": 8
    },
    "id": 2
  },
  "brand": {
    "id": 5
  },
  "model": {
    "id": 963
  },
  "modification": "V12 AMG B-turbo",
  "body": {
    "id": 6
  },
  "doors": 2,
  "gearbox": {
    "id": 1
  },
  "color": {
    "id": 10,
    "metallic": true
  },
  "year": 2016,
  "mileage": 32,
  "price": {
    "value": 80000,
    "currency": {
      "id": 1
    }
  },
  "post": {
    "auctions": true,
    "exchanges": {
      "type": {
        "id": 1
      },
      "payment": {
        "id": 2
      }
    }
  },
  "comments": {
    "allowed": true,
    "check": false
  }
},

```

```

"spareParts": false,
"user": {
  "id": 7069830,
  "type": "private"
},
"dates": {
  "created": "2017-09-13T06:26:35.000Z",
  "updated": "0000-00-00 00:00:00",
  "expired": "0000-00-00 00:00:00",
  "sold": "0000-00-00 00:00:00"
},
"status": {
  "id": 1
},
"partner": {
  "id": 1
},
"ip": 8091,
"description": {
  "ru": "Авто в идеальном состоянии, вложений не требует",
  "uk": "Авто в ідеальному стані, вкладень не потребує"
},
"region": {
  "id": 10
},
"city": {
  "id": 10
},
"moderated": true,
"damage": false,
"custom": false,
"photos": {
  "main": {
    "id": 196054982
  }
},
"credit": false,
"video": {
  "key": "lLEiT9PeHSg"
},
"categories": {
  "main": {
    "id": 1
  },
  "all": [
    {
      "id": 1
    }
  ]
},
"drive": {

```

```

        "id": 2
    },
    "VIN": "JKBVNCB164A66XXX",
    "confiscated": false,
    "verified": {
        "serviceStation": false
    },
    "seats": 2,
    "power": {
        "hp": 585,
        "kW": 430.56
    },
    "engine": {
        "volume": {
            "cc": 3500,
            "liters": 3.5
        }
    }
}

```

Точка входу `web/index.php`, з якої починається старт клієнтської частини в Yii 2, має наступний вигляд:

```

<?php

defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

require __DIR__ . '/../../vendor/autoload.php';
require __DIR__ . '/../../vendor/yiisoft/yii2/Yii.php';
require __DIR__ . '/../../common/config/bootstrap.php';
require __DIR__ . '/../config/bootstrap.php';

$config = yii\helpers\ArrayHelper::merge(
    require __DIR__ . '/../../common/config/main.php',
    require __DIR__ . '/../../common/config/main-local.php',
    require __DIR__ . '/../config/main.php',
    require __DIR__ . '/../config/main-local.php'
);

(new yii\web\Application($config))->run();

```

Висновок до другого розділу

В цьому розділі було розроблено та описано діаграми класів, варіантів використання, активності.

База даних, реалізована відповідно до вимог реляційної моделі, має стабільний, швидкий і постійний доступ до даних, необхідних для реалізації сайту.

Було розроблено та детально описано алгоритм процесів сайту, для реалізації веб додатку та всього необхідного функціоналу

РОЗДІЛ 3. ІНТЕРФЕЙС ТА АРХІТЕКТУРА ВЕБ - ДОДАТКУ

3.1 Архітектура та системні вимоги до веб – додатку

Цей програмний продукт створено з використанням фреймворку Yii2 (advanced). Тому архітектура є класичною для цього фреймворку.

Основа цієї архітектури будується шляхом поділу всього проекту на клієнтську та серверну частини.

Yii 2 використовує патерн Model – View – Controller, в якому:

- Model – модель даних
- View – представлення
- Controller – контролер

Веб – сервіс об'яв з продажу авто з можливістю порівняння ціни не вимагає спеціального апаратного забезпечення, додаткових засобів чи налаштувань для розгортки проекту.

Системні вимоги:

- 512 ГБ оперативної пам'яті
- 1ГБ вільного місця на жорсткому диску
- Процесор із тактовою частотою мінімум 1ГГц
- Будь яка операційна система
- Apache HTTP server версії 2.5 або вище
- PHP 7.0 або вище;
- Composer 2.0.

Для успішного розгортання проекту на сервері потрібно:

1. Переконайтеся, що сервер відповідає наведеним вище системним вимогам;
2. Здійснити операцію клонування проекту з репозиторію
3. Зайти в директорію проекту
4. Створити базу даних з назвою «car_ads_db»
5. Імпортувати базу даних з файлу «car_ads_db.sql»
6. Виконати команду «composer update»

7. Виконати переведення проекту в режим Production, для цього у файлі index.php потрібно замінити рядок

```
«defined('YII_DEBUG') or define('YII_DEBUG', true);  
defined('YII_ENV') or define('YII_ENV', 'dev');»
```

на

```
«defined('YII_DEBUG') or define('YII_DEBUG', false);  
defined('YII_ENV') or define('YII_ENV', 'prod');»;
```

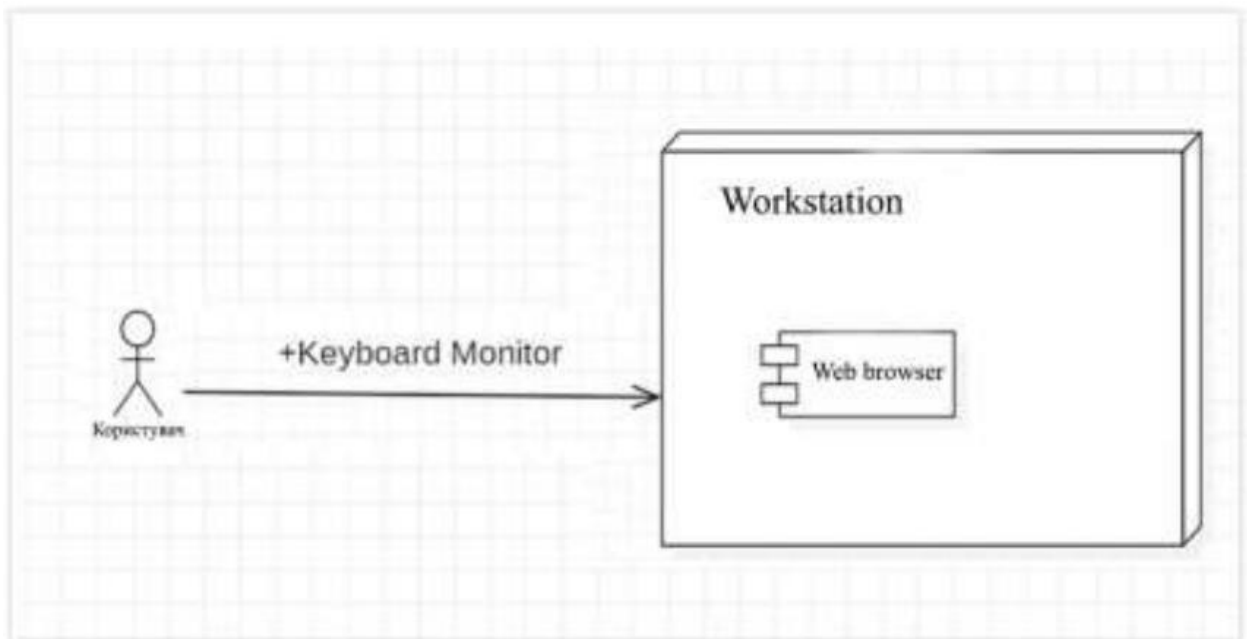


Рис. 3.1 – Діаграма розгортання веб – додатку

По діаграмі (Рисунок 3.1), що розгортання не вимагає від користувача ніяких додаткових дій, оскільки елементи платформи вже містять всі необхідні дані.

3.2 Структура інтерфейсу та порядок робити з веб - сервісом

При заході на веб – сервіс, користувач побачить головну сторінку сайту, на якій він може перейти на сторінку авторизації, реєстрації або ж скористатися пошуком об’яв на головній сторінці.

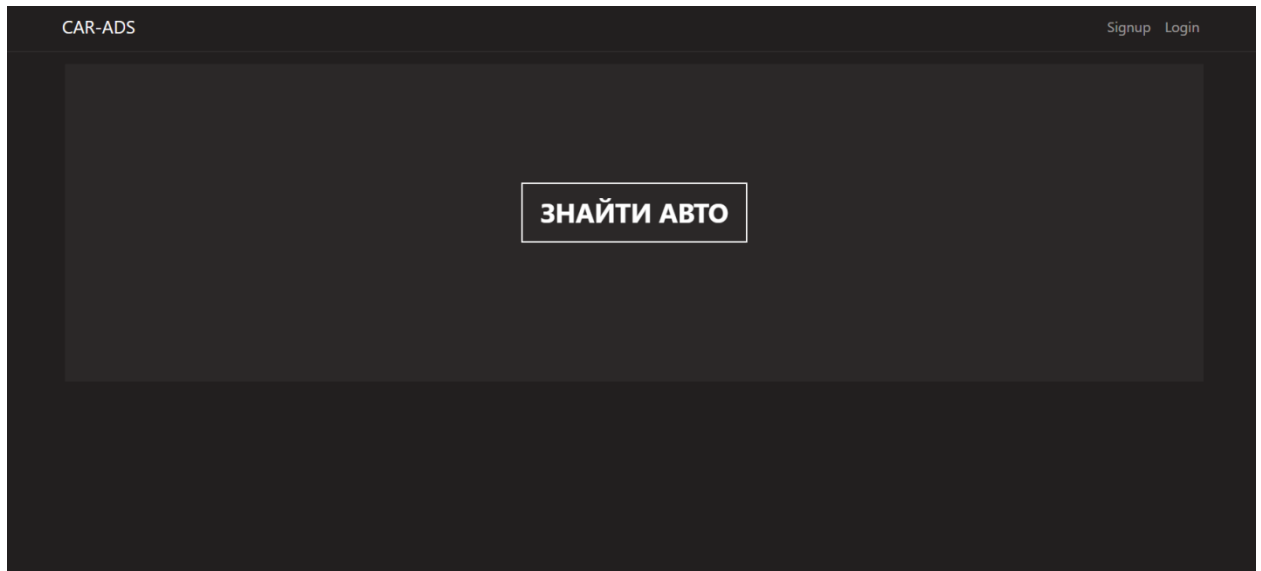


Рис. 3.2 – Головна сторінка

Слайдер на головній сторінці є фільтром для пошуку, тому після натискання кнопки «ЗНАЙТИ АВТО», користувач побачить основні фільтри пошуку.

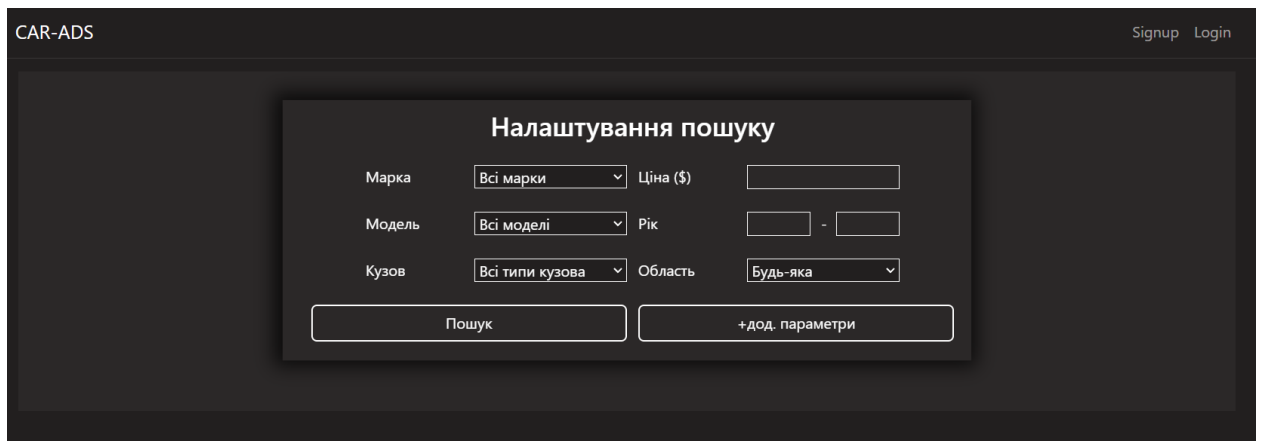


Рис. 3.3 – Фільтри пошуку

В даній секції користувач може вибрати основні фільтри пошуку для того щоб знайти саме те авто яке він і хотів. Користувач може обрати:

- Марку авто
- Модель авто
- Тип кузова
- Максимальну ціну(в доларах)
- Допустимий діапазон року випуску авто(від, до)
- Бажана область

The screenshot shows a web interface for 'CAR-ADS'. In the top right corner, there are links for 'Signup' and 'Login'. The main content area is titled 'Додаткові параметри' (Additional parameters). It contains several form elements:

- Двигун** (Engine): A range selector with 'від' (from) and 'до' (to) dropdowns.
- КПП** (Transmission): A dropdown menu with the value 'Будь-який' (Any).
- Паливо** (Fuel): A dropdown menu with the value 'Будь-яке' (Any).
- Пробіг (тис.км.)** (Mileage in thousands of km): An empty input field.
- Стан** (Condition): A dropdown menu with the value 'Будь-який' (Any).
- Привід** (Drive): A dropdown menu with the value 'Будь-який' (Any).

 At the bottom of the section are two buttons: 'Пошук' (Search) and 'Назад' (Back).

Рис. 3.4 – Додаткові параметри пошуку

Якщо користувач хоче більш детально описати авто, яке він шукає, то він може натиснути на кнопку «+доп. параметри». У даній секції користувач може вказати:

- Припустимий об'єм двигуну(від, до)
- Вид палива
- Теперішній стан авто
- Тип КПП(коробки передач)
- Привід авто

Якщо користувач натисне на кнопку пошуку в секції «Налаштування пошуку», то будуть враховуватись фільтри тільки з цієї секції, якщо в секції «Додаткові параметри», то будуть враховуватись всі фільтри з обох секцій при пошуку.

Після натискання на кнопку пошуку, на сторінці, без перезавантаження, з'явиться секція «Результати пошуку».

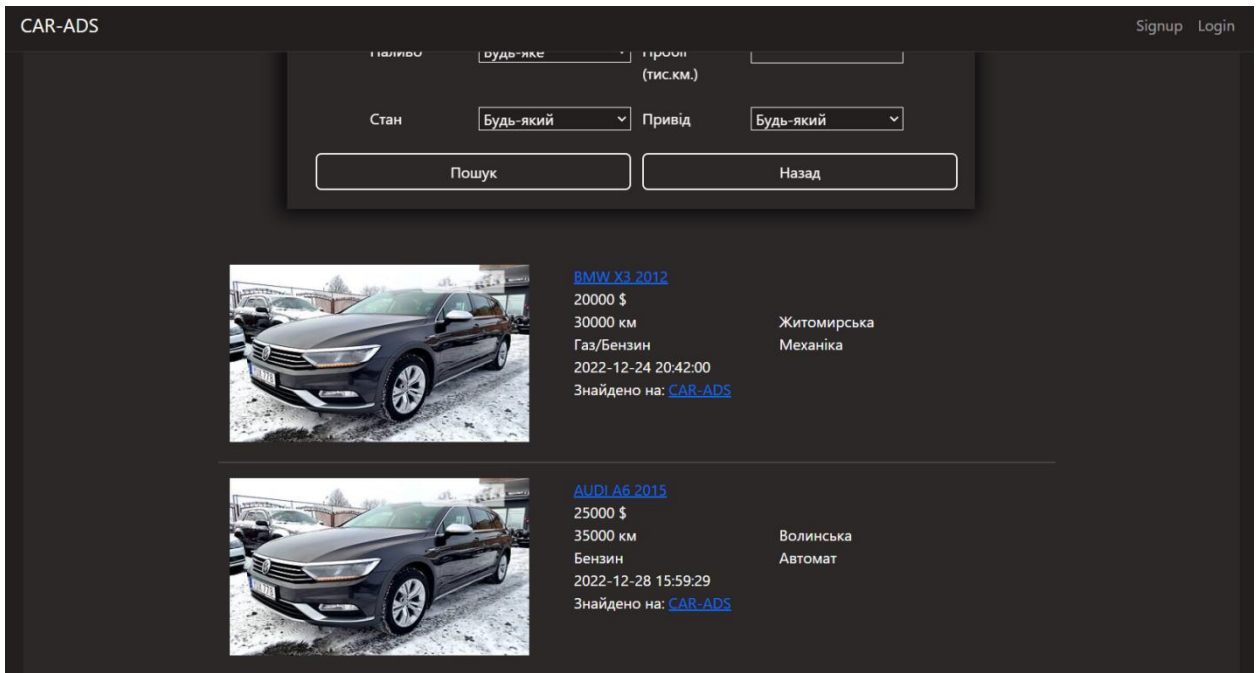


Рис. 3.5 – Результати пошуку

Для повної взаємодії з веб – сервісом користувачу потрібно пройти авторизацію на сайті.

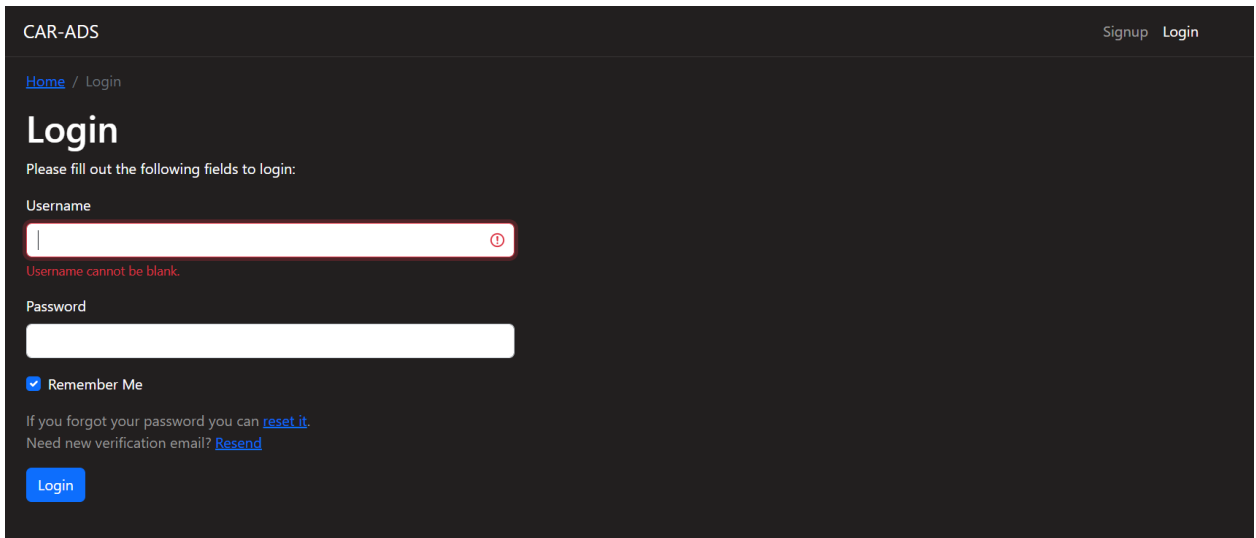


Рис. 3.6 – Сторінка авторизації

Якщо користувач ще не зареєстрований, то він може пройти реєстрацію на сторінці реєстрації.



Рис. 3.7 – Сторінка реєстрації

3.3 Тестування роботи веб – додатку

Первинне тестування веб - додатку було виконано за допомогою YII 2, стандартного інструменту налагодження. Цей інструмент відстежує наступні показники:

- Час виконання запитів до бази даних
- Час завантаження
- Кількість затраченої пам’яті
- Статус код

Дані тестування розміщені у таблиці 3.1

Таблиця 3.1

№	Назва сторінки	Статус код	Час завантаження	Кількість затраченої пам’яті	Час виконання запитів до бази даних
1	Головна сторінка	200	215мс	8.01МБ	123мс
2	Сторінка оголошення	200	134мс	7.53МБ	82мс
3	Сторінка авторизації	200	93мс	7.12МБ	20мс

Веб - сервіс також був протестований на адаптивність на різних девайсах з різною роздільною здатністю екрану. Ці тести виявили всі проблеми та успішно їх вирішили. Це дозволить використовувати сайт на різних пристроях, від мобільних телефонів до комп'ютерів.

Висновок до третього розділу

У цьому розділі обговорюється архітектура веб-додатку, описуються використані архітектурні патерни та визначаються ресурси, необхідні для розгортання сайту на сервері. Також розглядається структура інтерфейсу веб-додатку та його робочі процедури. Детально описано основні сторінки веб-сайту, а також алгоритми роботи.

Веб - додаток було протестовано. Будь-які помилки, виявлені під час тестування, були вчасно виправлені. Тестування проводилося за такими параметрами, як час завантаження сторінки, обсяг використаної пам'яті та час виконання запитів до бази даних. В результаті було зроблено висновок, що веб-сайт працює належним чином і відповідає всім зазначеним вимогам.

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи магістра було спроектовано та реалізовано веб – сервіс з продажу авто з можливістю порівняння ціни, для цього було вирішено такі завдання:

- Визначивши місію, проаналізувавши тематичну сферу та проаналізувавши існуючі проекти-аналоги, можна проаналізувати ключові моменти та поєднати найкраще з кожної аналогії.
- База даних реалізована відповідно до вимог реляційної моделі, що забезпечує безперервний та стабільний доступ до даних.
- Розглянуто алгоритми веб - додатку, аналіз та розробка взаємодії між функціональними класами веб-сайту.
- Розглянуто архітектуру веб – додатку, необхідні умови для розгортання, структуру інтерфейсу. Проведено тестування роботи веб – сервісу.

ПЕРЕЛІК ВИКОРСИТАНОЇ ЛІТЕРАТУРИ

- 1) Офіційна сторінка фреймворку Yii 2 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.yiiframework.com>
- 2) Офіційна сторінка веб – сервісу оголошень OLX [Електронний ресурс] – Режим доступу до ресурсу: <https://www.olx.ua/uk/>
- 3) Офіційна сторінка веб – сервісу оголошень з продажу авто AUTO RIA [Електронний ресурс] – Режим доступу до ресурсу: <https://auto.ria.com/uk/>
- 4) Офіційна сторінка інституту досліджень авторинку [Електронний ресурс] – Режим доступу до ресурсу: <https://eauto.org.ua/>
- 5) Документ дослідження попиту на авто [Електронний ресурс] – Режим доступу до ресурсу: <https://olshansky.ua/wp-content/uploads/2021/09/ukr>
- 6) Офіційна сторінка мови програмування PHP [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/>
- 7) Офіційна сторінка MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/>
- 8) Офіційна сторінка Open Server [Електронний ресурс] – Режим доступу до ресурсу: <https://ospanel.io/>

ДОДАТКИ

```
<?php
```

```
namespace frontend\controllers;

use frontend\models\CarInfo;
use frontend\models\CarModel;
use frontend\models\CarState;
use frontend\models\CarWheelDrive;
use frontend\models\Regions;
use frontend\models\BodyType;
use frontend\models\EngineType;
use frontend\models\FuelType;
use frontend\models\ResendVerificationEmailForm;
use frontend\models\TradeMark;
use frontend\models\TransmissionType;
use frontend\models\VerifyEmailForm;
use PHPUnit\Util\Log\JSON;
use Yii;
use yii\base\InvalidArgumentException;
use yii\db\Connection;
use yii\web\BadRequestHttpException;
use yii\web\Controller;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use common\models\LoginForm;
use frontend\models>PasswordResetRequestForm;
use frontend\models\ResetPasswordForm;
use frontend\models\SignupForm;
use frontend\models>ContactForm;
```

```

/**
 * Site controller
 */
class SiteController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['logout', 'signup'],
                'rules' => [
                    [
                        'actions' => ['signup'],
                        'allow' => true,
                        'roles' => ['?'],
                    ],
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [

```

```

        'logout' => ['post'],

    ],

],

];

}

/**
 * {@inheritdoc}
 */
public function actions()
{
    return [
        'error' => [
            'class' => 'yii\web\ErrorAction',
        ],
        'captcha' => [
            'class' => 'yii\captcha\CaptchaAction',
            'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
        ],
    ];
}

/**
 * Displays homepage.
 *
 * @return mixed
 */
public function actionIndex()
{
    $bodyTypes = BodyType::find()->all();

```

```

$engineTypes = EngineType::find()->all();
$transmissionTypes = TransmissionType::find()->all();
$fuelTypes = FuelType::find()->all();
$tradeMark = TradeMark::find()->all();
$carState = CarState::find()->all();
$carWheelDrive = CarWheelDrive::find()->all();
$regions = Regions::find()->all();
if (Yii::$app->request->isPost) {
    if (isset($_POST['sval'])) {
        $data = $_POST['sval'];
        $tradeMark = TradeMark::find()
            ->where(['name' => $data])
            ->one();
        $carModels = CarModel::find()
            ->where(['trade_mark_id' => $tradeMark['id']])
            ->all();
        $carModelsArr = [];
        foreach ($carModels as $carModel) {
            array_push($carModelsArr, $carModel['name']);
        }
        return json_encode($carModelsArr);
    }
    if ($_POST['pass'] == 'search') {
        $select = 'SELECT * FROM carInfo WHERE id > 0';
        if ($_POST['mark'] != 'all') {
            $car_mark = TradeMark::find()
                ->where(['name' => $_POST['mark']])
                ->one();
        }
    }
}

```

```

        $select = $select . ' AND ' . 'trade_mark_id = ' .
$car_mark['id'];
    }
    if ($_POST['model'] !== "all") {
        $car_model = CarModel::find()
            ->where(['name' => $_POST['model']])
            ->one();
        $select = $select . ' AND ' . 'model_name_id = ' .
$car_model['id'];
    }
    if ($_POST['price'] !== 'all') {
        $select = $select . ' AND ' . 'price >= ' .
intval($_POST['price']);
    }
    if ($_POST['year_from'] !== 'all') {
        $select = $select . ' AND ' . 'year >= ' .
intval($_POST['year_from']);
    }
    if ($_POST['year_to'] !== 'all') {
        $select = $select . ' AND ' . 'year <= ' .
intval($_POST['year_to']);
    }
    if ($_POST['body'] !== 'all') {
        $car_body = BodyType::find()
            ->where(['name' => $_POST['body']])
            ->one();
        $select = $select . ' AND ' . 'body_type_id = ' .
$car_body['id'];
    }
    if ($_POST['region'] !== 'all') {
        $car_region = Regions::find()
            ->where(['name' => $_POST['region']])

```

```

        ->one();

        $select = $select . ' AND ' . 'region_id = ' .
$car_region['id'];

        $command = Yii::$app->db->createCommand($select);

        $car_add = $command->queryAll();

        return $car_add[0]['price'];
    }

    if ($_POST['engine_from'] !== 'all') {
        $engineTypes = EngineType::find()
            ->where(['name' => $_POST['engine_from']])
            ->one();

        $select = $select . ' AND ' . 'engine_type_id >= ' .
$engineTypes['id'];
    }

    if ($_POST['engine_to'] !== 'all') {
        $engineTypes = EngineType::find()
            ->where(['name' => $_POST['engine_to']])
            ->one();

        $select = $select . ' AND ' . 'engine_type_id <= ' .
$engineTypes['id'];
    }

    if ($_POST['trans'] !== 'all') {
        $trans = TransmissionType::find()
            ->where(['name' => $_POST['trans']])
            ->one();

        $select = $select . ' AND ' . 'transmission_id = ' .
$trans['id'];
    }

    if ($_POST['fuel'] !== 'all') {
        $fuel = FuelType::find()
            ->where(['name' => $_POST['fuel']])

```

```

        ->one();
        $select = $select . ' AND ' . 'fuel_type_id = ' .
$fuel['id'];
    }
    if ($_POST['distance'] !== 'all') {
        $select = $select . ' AND ' . 'mile_age <= ' .
intval($_POST['distance']);
    }
    if ($_POST['state'] !== 'all') {
        $state = CarState::find()
            ->where(['name' => $_POST['state']])
            ->one();
        $select = $select . ' AND ' . 'car_state_id = ' .
$state['id'];
    }
    if ($_POST['wheel'] !== 'all') {
        $wheel = CarWheelDrive::find()
            ->where(['name' => $_POST['wheel']])
            ->one();
        $select = $select . ' AND ' . 'car_wheel_drive_id = ' .
$wheel['id'];
    }
    $command = Yii::$app->db->createCommand($select);
    $car_adds = $command->queryAll();

    if (count($car_adds) > 0) {
        for($i = 0; $i < count($car_adds); $i++) {
            $car_mark = TradeMark::find()
                ->where(['id' => $car_adds[$i]['trade_mark_id']])
                ->one();
            $car_adds[$i]['trade_mark_id'] = $car_mark['name'];
        }
    }

```

```

$car_model = CarModel::find()
    ->where(['id' => $car_adds[$i]['model_name_id']])
    ->one();
$car_adds[$i]['model_name_id'] = $car_model['name'];
$regions = Regions::find()
    ->where(['id' => $car_adds[$i]['region_id']])
    ->one();
$car_adds[$i]['region_id'] = $regions['name'];
$car_fuel = FuelType::find()
    ->where(['id' => $car_adds[$i]['fuel_type_id']])
    ->one();
$car_adds[$i]['fuel_type_id'] = $car_fuel['name'];
$engine_volume = EngineType::find()
    ->where(['id' =>
$car_adds[$i]['engine_type_id']])
    ->one();
$car_adds[$i]['engine_type_id'] =
$engine_volume['name'];
$trans = TransmissionType::find()
    ->where(['id' =>
$car_adds[$i]['transmission_id']])
    ->one();
$car_adds[$i]['transmission_id'] = $trans['name'];
}
$json = json_encode($car_adds);
return $json;
} else return 'Нічого не знайдено';
}
}
return $this->render('index', [
    'bodyTypes' => $bodyTypes,

```

```

        'engineTypes' => $engineTypes,
        'transmissionTypes' => $transmissionTypes,
        'fuelTypes' => $fuelTypes,
        'tradeMark' => $tradeMark,
        'carState' => $carState,
        'carWheelDrive' => $carWheelDrive,
        'regions' => $regions,
    ]);
}

public function actionPost($id) {
    $car_add = CarInfo::find()
        ->where(['id' => $id])
        ->one();

    $car_mark = TradeMark::find()
        ->where(['id' => $car_add['trade_mark_id']])
        ->one();

    $car_add['trade_mark_id'] = $car_mark['name'];

    $car_model = CarModel::find()
        ->where(['id' => $car_add['model_name_id']])
        ->one();

    $car_add['model_name_id'] = $car_model['name'];

    $regions = Regions::find()
        ->where(['id' => $car_add['region_id']])
        ->one();

    $car_add['region_id'] = $regions['name'];

    $car_fuel = FuelType::find()
        ->where(['id' => $car_add['fuel_type_id']])
        ->one();

    $car_add['fuel_type_id'] = $car_fuel['name'];
}

```

```

$engine_volume = EngineType::find()
    ->where(['id' => $car_add['engine_type_id']])
    ->one();
$car_add['engine_type_id'] = $engine_volume['name'];
$trans = TransmissionType::find()
    ->where(['id' => $car_add['transmission_id']])
    ->one();
$car_add['transmission_id'] = $trans['name'];
$body = BodyType::find()
    ->where(['id' => $car_add['body_type_id']])
    ->one();
$car_add['body_type_id'] = $body['name'];
$state = CarState::find()
    ->where(['id' => $car_add['car_state_id']])
    ->one();
$car_add['car_state_id'] = $state['name'];
$wheel = CarWheelDrive::find()
    ->where(['id' => $car_add['car_wheel_drive_id']])
    ->one();
$car_add['car_wheel_drive_id'] = $wheel['name'];

return $this->render('post');
}

/**
 * Logs in a user.
 *
 * @return mixed
 */
public function actionLogin()
{

```

```

        if (!Yii::$app->user->isGuest) {
            return $this->goHome();
        }

        $model = new LoginForm();
        if ($model->load(Yii::$app->request->post()) && $model->login()) {
            return $this->goBack();
        }

        $model->password = '';

        return $this->render('login', [
            'model' => $model,
        ]);
    }

    /**
     * Logs out the current user.
     *
     * @return mixed
     */
    public function actionLogout()
    {
        Yii::$app->user->logout();

        return $this->goHome();
    }

    /**
     * Displays about page.

```

```

*
* @return mixed
*/
public function actionAbout()
{
    return $this->render('about');
}

/**
 * Signs user up.
 *
 * @return mixed
 */
public function actionSignup()
{
    $model = new SignupForm();
    if ($model->load(Yii::$app->request->post()) && $model->signup()) {
        Yii::$app->session->setFlash('success', 'Thank you for
registration. Please check your inbox for verification email.');
```

return \$this->goHome();

```

    }

    return $this->render('signup', [
        'model' => $model,
    ]);
}

/**
 * Requests password reset.
 *
```

```

* @return mixed
*/
public function actionRequestPasswordReset()
{
    $model = new PasswordResetRequestForm();
    if ($model->load(Yii::$app->request->post()) && $model->validate()) {
        if ($model->sendEmail()) {
            Yii::$app->session->setFlash('success', 'Check your email for
further instructions.');
```

return \$this->goHome();

```
        }

        Yii::$app->session->setFlash('error', 'Sorry, we are unable to
reset password for the provided email address.');
```

return \$this->render('requestPasswordResetToken', [
 'model' => \$model,
]);

```
    }

/**
 * Resets password.
 *
 * @param string $token
 * @return mixed
 * @throws BadRequestHttpException
 */
public function actionResetPassword($token)
{

```

```

    try {
        $model = new ResetPasswordForm($token);
    } catch (InvalidArgumentException $e) {
        throw new BadRequestHttpException($e->getMessage());
    }

    if ($model->load(Yii::$app->request->post()) && $model->validate() &&
$model->resetPassword()) {
        Yii::$app->session->setFlash('success', 'New password saved.');
```



```

        return $this->goHome();
    }

    return $this->render('resetPassword', [
        'model' => $model,
    ]);
}

/**
 * Verify email address
 *
 * @param string $token
 * @return yii\web\Response
 * @throws BadRequestHttpException
 */
public function actionVerifyEmail($token)
{
    try {
        $model = new VerifyEmailForm($token);
    } catch (InvalidArgumentException $e) {

```

```

        throw new BadRequestHttpException($e->getMessage());
    }

    if (($user = $model->verifyEmail()) && Yii::$app->user->login($user))
    {
        Yii::$app->session->setFlash('success', 'Your email has been
confirmed!');
        return $this->goHome();
    }

    Yii::$app->session->setFlash('error', 'Sorry, we are unable to verify
your account with provided token.');
```

```

    return $this->goHome();
}

/**
 * Resend verification email
 *
 * @return mixed
 */
public function actionResendVerificationEmail()
{
    $model = new ResendVerificationEmailForm();
    if ($model->load(Yii::$app->request->post()) && $model->validate()) {
        if ($model->sendEmail()) {
            Yii::$app->session->setFlash('success', 'Check your email for
further instructions.');
```

```

            return $this->goHome();
        }

        Yii::$app->session->setFlash('error', 'Sorry, we are unable to
resend verification email for the provided email address.');
```

```

    }
}

```

```
        return $this->render('resendVerificationEmail', [
            'model' => $model
        ]);
    }
}
```