

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

Дворніков Дмитро Юрійович

**РОЗРОБКА ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ
ДІЯЛЬНОСТІ АВТОСАЛОНА**

**кваліфікаційна робота
здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми «Комп’ютерні науки та інформаційні технології»
за спеціальністю 122 Комп’ютерні науки**

Особистий підпис _____ Дмитро ДВОРНІКОВ

Науковий керівник _____ Юрій КОЗУБ, д.т.н., професор

Завідувач кафедри _____ Юрій КОЗУБ, д.т.н., професор

Міністерство освіти і науки України
Державний заклад „Луганський національний університет
імені Тараса Шевченка”

Інститут математики та інформаційних технологій	
Кафедра	Математики та інформатики
Освітній ступень	Бакалавр
Напрямок підготовки (спеціальність)	122 «Комп’ютерні науки» (код, назва)
Галузь знань	12 «Інформаційні технології» (код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри МІ

Ю. Г. Козуб

(підпис)

(ініціали, прізвище)

“ ” 2025 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Дворнікова Дмитра Юрійовича

(прізвище, ім’я, по батькові)

1. Тема Розробка веборієнтованої інформаційної системи для підтримки діяльності автосалона

Керівник кваліфікаційної роботи

Козуб Ю.Г.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету

від“ ” 2025 року №

2. Строк подання студентом проекту (роботи)

3. Вихідні дані до роботи (проекту)

у результаті виконання роботи

повинна бути детально описана і розроблена веборієнтована система для підтримки діяльності автосалона

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об’єкт розробки)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розробка технічного завдання на створення веборієнтованої системи для підтримки діяльності автосалона

Вибір інструментів для написання майбутньої програми

Розробка інформаційного забезпечення веб-орієнтованої системи

Розробка інформаційного забезпечення веб-орієнтованої системи

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Консультанти розділів проекту (роботи)

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

7. Дата видачі завдання „_____” _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	до 1 лютого	
2	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	до 20 березня	
3	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	до 1 квітня	
4	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	до 15 квітня	
5	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	до 30 квітня	
6	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	до 15 травня	
7	Попередній захист роботи на кафедрі	До 30 травня	
8	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
9	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Здобувач вищої освіти

підпис

Дмитро ДВОРНИКОВ

(ініціали, прізвище)

Керівник проекту (роботи)

підпис

Юрій КОЗУБ

АНОТАЦІЯ

Дворніков Д.Ю.

Тема: Розробка веборієнтованої системи для підтримки діяльності автосалона.

Спеціальність: 122 „Комп’ютерні науки”

Установа: ДЗ ЛНУ імені Т.Шевченка, 2025р.

Кваліфікаційна робота містить: 71 с., 36 рис., 4 табл., 3 додат., 32 джерела.

Об’єкт дослідження – веб-орієнтована система.

Предмет дослідження – технологія розробки клієнт-серверних CRUD систем.

Мета роботи - розробка інформаційної система для підтримки діяльності автосалону, що дозволяє здійснювати заходи з підбору та продажу автомобілів обраних марок та моделей.

Результат роботи. Продукт, що складається із сайту з сучасним дизайном та інтуїтивно-зрозумілим інтерфейсом, що дозволяє менеджерів ефективно співпрацювати з клієнтами, бо, по-перше, містить доволі великий каталог автомобілів, а по-друге, може безпосередньо вносити деякі зміни у поєднану реляційну базу даних через відповідну клієнт-серверну частину, функціонал якої може відрізнятися в залежності від прав того чи іншого користувача.

Висновки. Інформаційна система складається з багатосторінкового сайту(HTML+CSS), клієнтської та серверної частин, що написані на JavaScript, а також реляційної бази даних на мові SQL.

Ключові слова: сайт, інформаційна система, реляційна база даних, JavaScript, SQL.

ABSTRACT

Dvornikov, D.

Theme: Development of a web-based system to support the activities of an autohouse.

Specialty: 122 "Computer Science"

Institution: Taras Shevchenko Luhansk National University, 2025.

Qualification work contains: 71 pages, 36 figures, 4 tables, 3 appendices, 32 sources.

Object of research: web-oriented system.

Subject of research: technology of development of client-server CRUD systems.

Purpose of the study: development of information system to support the activities of an autohouse, which allows to carry out activities on selection and sale of cars of selected brands and models.

Result of research. The product consists of a site with a modern design and an intuitive interface that allows the manager to effectively collaborate with clients, because, firstly, it contains a fairly large catalog of cars, and secondly, it can directly make some changes to the combined relational database through the corresponding client-server part, the functionality of which may differ depending on the rights of a particular user.

Conclusions. The information system consists of a multi-page site (HTML+CSS), client and server parts written in JavaScript, as well as a relational database in the SQL language.

Keywords: site, information system, relational database, JavaScript, SQL.

ЗМІСТ

Анотація	5
Вступ.....	9
1 Розробка технічного завдання на створення веб-орієнтованої системи підтримки діяльності автосалону	10
1.1 Обґрунтування актуальності використання веб-орієнтованих інформаційних систем	10
1.2 Огляд існуючих аналогів та формування концепції для вирішення поставленого завдання.....	11
Висновок по розділу 1	14
2 Вибір інструментів для написання майбутньої програми	15
2.1 Особливості веб-технологій для розробки інформаційних систем	15
2.2 Існуючі рішення у веб-технології та їх призначення	15
2.3 Вибір інструменту для розробки веб-орієнтованої системи автосалону	20
2.3.1 React.js та його основні складові.....	24
2.3.2 Що таке Node.js і як вона влаштована	27
2.4 Вибір типу СУБД	29

Висновок по розділу 2	31
3 Розробка інформаційного забезпечення веб-орієнтованої системи	33
3.1 Складання концептуальної схеми предметної галузі.....	33
3.2 Побудова даталогічної схеми бази даних.....	36
3.3 Характеристики різних СУБД SQL.....	39
Висновок по розділу 3	45
4 Розробка програмного забезпечення програмного комплексу.....	46
4.1 Підготовка до реалізації, налаштування необхідних утиліт та компонентів	46
4.2 Завантаження бази даних «Автосалон» у MySQL.....	47
4.1 Проектування та розробка сайту	49
4.2 Створення структури та опис React-компонентів панелі адміністратора.....	56
4.3 Забезпечення зв'язку між компонентами шляхом використання стандартної бібліотеки маршрутизації React - React Router	58
4.4 Конфігурування складових компонентів.....	59
Висновок по розділу 4	63
5 Висновок	64
6 Список використаних джерел	65
Додаток А.....	68
Додаток Б.....	69
Додаток В	71

ВСТУП

Підтримка діяльності автосалону – це комплексне рішення, яке дозволяє упорядкувати наявні модельні пропозиції та надати зазначити їх характеристики, забезпечити зворотній зв'язок від клієнта, запропонувати йому ряд послуг та проконтролювати їх виконання – розв'язанням поставлених проблем може виступити спеціально розроблена веб-орієнтована система.

Зазначена система повинна спиратися на перевірені програмні рішення задля зручності та надійності операцій з даними. Для цього підійде класична клієнт-серверна архітектура, що поєднана з базою даних.

Тому комплексному визначенню завдань щодо побудови необхідної веб-орієнтованої системи підтримки діяльності автосалону через співставний аналіз наданих нам вимог та ознайомлення з вже існуючими системами задля вибору найбільш привабливого для нас варіанту, а також пошуку та вибору необхідного інструментів та визначення з типом бази даних буде й присвячений даний звіт, який, у свою чергу, буде прологом до реалізації вже безпосередньо дипломної роботи.

1 РОЗРОБКА ТЕХНІЧНОГО ЗАВДАННЯ НА СТВОРЕННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ПІДТРИМКИ ДІЯЛЬНОСТІ АВТОСАЛОНУ

1.1. Обґрунтування актуальності використання веб-орієнтованих інформаційних систем

На сьогоднішній день важко уявити наше життя без усіляких програмних додатків, інтернет-магазинів, електронних довідників та ін. Обробка великих об'ємів інформації, її зберігання чи корегування, процес формування звітності чи сам процес життєдіяльності будь-якої організації неможливо уявити без роботи з базами даних та поєднаною з нею клієнт-серверною архітектурою. Для цих цілей – CRUD-операцій та зручної взаємодії менеджера та клієнта, структурування даних та їх зрозумілого відображення - ідеально підходить веб-орієнтована система[13].

Однак, виникає логічне питання про те, яку технологію використовувати, щоб втілити в життя необхідний функціонал та задовольнити всі поставлені вимоги. Тут доцільно розглянути можливість створення веб-орієнтованої програми[4].

Його відмінні риси видно в наступному:

- Кросплатформність. Як клієнт виступає веб-браузер, який входить до складу будь-якої операційної системи. Оновлення та супровід браузера лежить на його розробнику.

- Мобільність. Працювати із системою ви можете, перебуваючи у будь-якому місці, де є Інтернет та з будь-якого пристрою, в якому є інтернет-браузер. Таким чином, користувач не обмежений вимогами до апаратної частини.

- Низька сукупна вартість володіння. Вартість володіння веб-орієнтованою системою фактично укладена у розробці, підтримці та розвитку серверної частини (веб-сервера та сервера додатків) та володінні сервером бази даних системи.



Рис. 1.1 – Загальна структура веб-програми

Значна кількість сучасних напрацювань та технологій присвячена Інтернету. Це можна легко довести. Озирнувшись довкола та проаналізувавши обстановку, в якій зараз живуть суспільства та держави, ми можемо побачити, що більший відсоток усіх банківських операцій, процес стягування податків та контроль інших сфер громадської діяльності, онлайн-продажі, а також підтримка розподілених транснаціональних компаній та здійснення документообігу дедалі тісніше переплітаються з веб-технологіями[8].

1.2 Огляд існуючих аналогів та формування концепції для вирішення поставленого завдання.

На сучасному ринку представлено доволі багато програмних рішень стосовно нашої проблематики. Тут є як просто рекламні сайти-візитки, так і цілі розгалужені системи, що підтримують безліч функцій новиноного порталу, дошки оголошень, майданчика дистриб'юторів різноманітних автовиробників, професійних оглядів новинок тощо. Нам потрібне більш медіанне рішення, бо у надпотужній системі немає потреби, але й доволі примітивні лендінги теж не підходять[8].

Почнемо аналізувати один із найпопулярніших сайтів про автомобілі – Infocar.ua. Тут представлені всі авто, які коли-небудь продавалися чи виготовлялися на території України. Також доступний форум, де власники чи

просто зацікавлені особи обмінюються думками, новини автосвіту, відео огляди авто, контакти автосалонів тощо.

The screenshot displays the Infocar.ua website interface. The top navigation bar includes the Infocar logo, a search bar, and links for 'Авто тижня' (CUPRA Terramar), 'Додати оголошення' (Add advertisement), and 'Вхід' (Login). Below this, a secondary navigation bar categorizes content into 'НОВІ АВТОМОБІЛІ', 'Б/У АВТО', 'ВІДГУКИ', 'ТЕСТ-ДРАЙВИ', 'НОВИНИ', 'ВІДЕО', 'МОТО', and 'РОЗМИТНЕННЯ'. The main content area is divided into sections for 'Нові автомобілі' (New cars) with filters for body types (седан, кросовер, хетчбек, купе, універсал, мінівен) and price ranges (15,000 \$ to 50,000 \$), 'Каталог моделей' (Model catalog) listing various brands, and 'Б/у автомобілі' (Used cars) with a search form for make, model, year, and price. A featured article titled '#ЩоПочому: НАЙПОТУЖНІШИЙ пікап. FORD RAPTOR. Ranger на максималках' is also visible. Below the main content, a browser address bar shows the URL 'volkswagen-passat.infocar.ua/passat-variant_id7004.html'. The bottom section of the image shows a detailed view of the 'Volkswagen Passat Variant 2023' page, featuring a large image of the car, its price range (\$40,475 - \$49,847), and a table of specifications.

Двигун	Мощність	Коробка	Привід	Версія	Ціна
1.5 eTSI MHEV	150 л.с.	7-DSG	2WD	Business	40.475 \$
2.0 TFSI	204 л.с.	7-DSG	2WD	Business	43.363 \$
2.0 TDI	150 л.с.	7-DSG	2WD	Business	45.441 \$

Рис. 1.2 – Загальний вигляд сайту Infocar.ua

Але й інші типи сайтів на автомобільну тематику, наприклад, сайти з продажу авто, наприклад auto.ria. Цей сайт може бути цікавим, перш за все, як розміщується

інформація про конкретний автомобіль, його комплектації тощо з метою подальшого продажу.

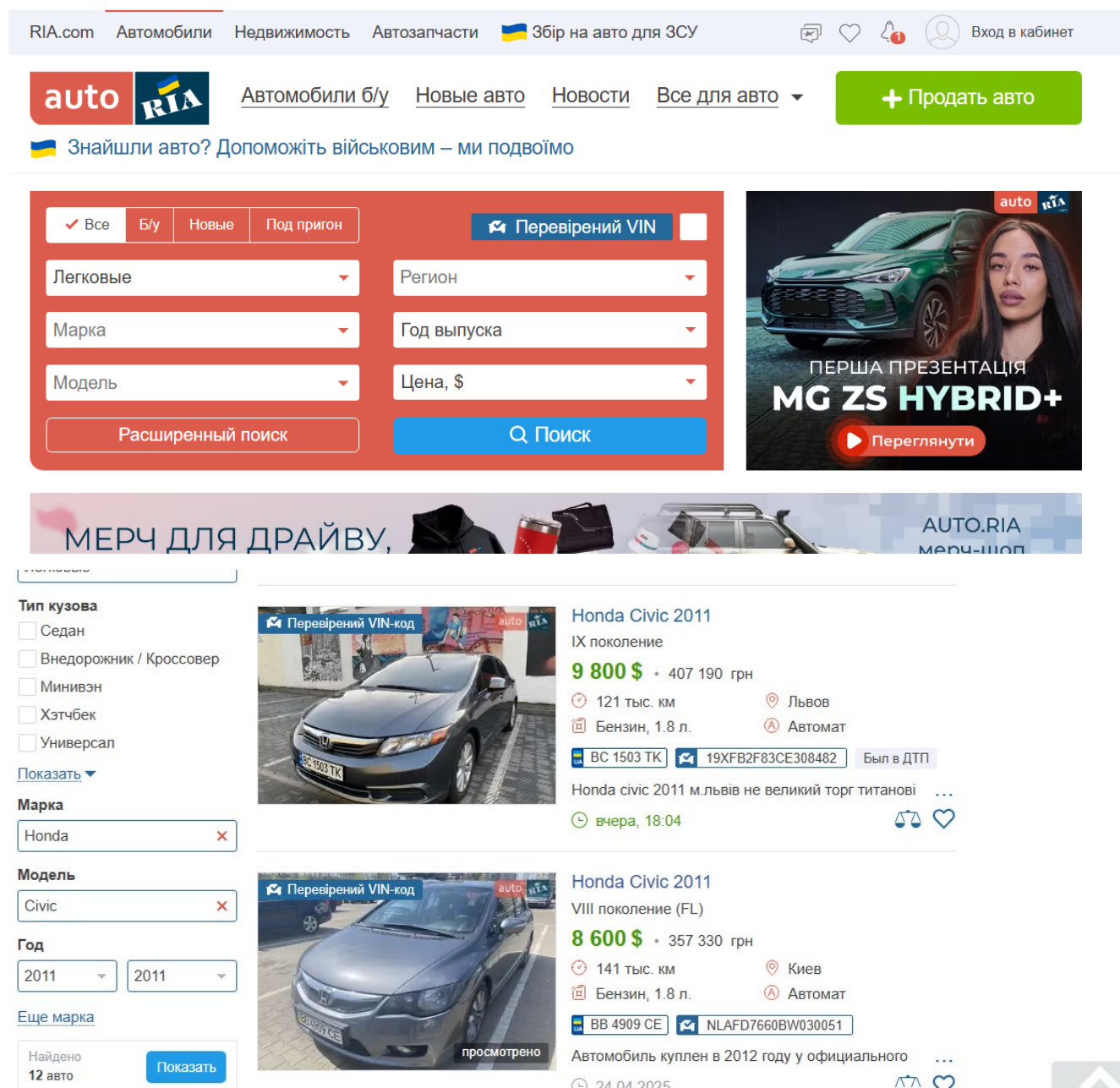


Рис. 1.3 – Загальний вигляд сайту auto.ria

Також можна розглянути сайт будь-якого комерційного автодилера, наприклад Hyundai [7; 24].

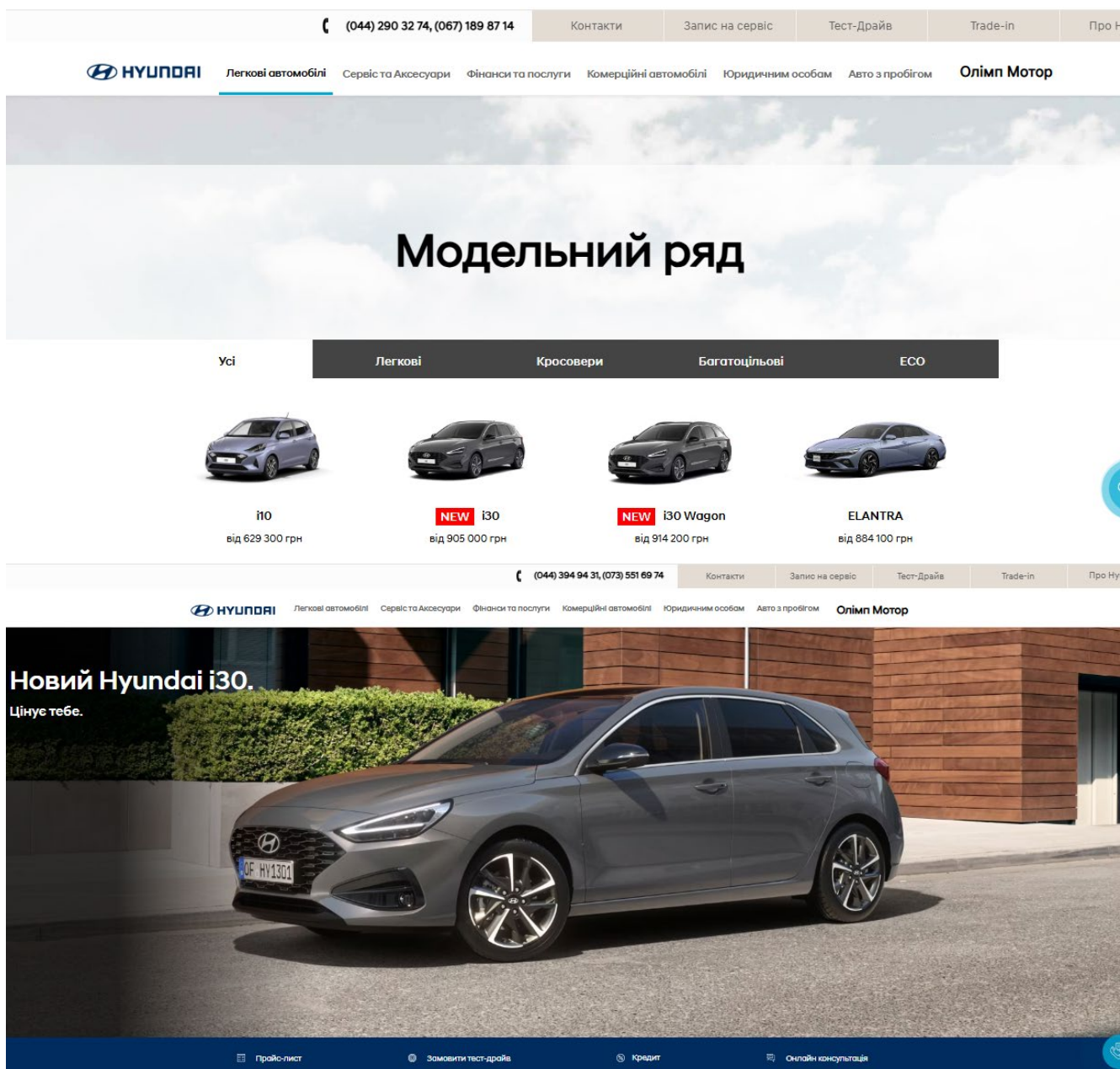


Рис. 1.4 – Загальний вигляд сайту olimpmotor.com.ua

Отже, у проміжному висновку, ми можемо зазначити, що окрім представлення автомобілів та опису їх технічних характеристик, необхідно ще й підтримувати зворотній зв'язок з клієнтом, забезпечити адміністрування як представлених пропозицій, так і збереження можливостей до додачі нових[8]

Висновок по розділу 1

Проаналізовані вже готові впровадження рішення з підтримки діяльності автосалону, виявлені необхідні компоненти та функціонал.

2. ВИБІР ІНСТРУМЕНТІВ ДЛЯ НАПИСАННЯ МАЙБУТНЬОЇ ПРОГРАМИ

2.1. Особливості веб-технологій для розробки інформаційних систем

Web-технології – комплекс технічних, комунікаційних, програмних методів вирішення завдань організації спільної діяльності користувачів із застосуванням Інтернету. Привабливість Web-технологій як засоби доставки інформації багато в чому визначає універсальний інтерфейс між людиною та комп'ютером. Кожній людині зрозумілі написи, заголовки, посилання, картинки.

Веб-інтерфейс як доступу до інформації інтуїтивно зрозумілий. Наслідком простоти веб-інтерфейсу є широке використання Інтернету як каналу комунікації. Браузер – програма для перегляду веб-сторінок та роботи з інформацією у веб-інтерфейсі. Браузери – програми, якими забезпечені всі майже всі сучасні електронні пристрої, так звані «гаджети». Теоретично всі браузери повинні відображати всі сайти, зроблені за стандартами, однаково, хоча на практиці є багато особливостей. Найбільш популярні браузери: Internet Explorer, Firefox, Opera, Safari, Chrome.

Інструмент для написання сайту дл підтримки діяльності автосалону повинен бути гнучким, швидко працювати і займати якнайменше ресурсів мережі та комп'ютера, а також мати можливість ефективно взаємодіяти із зовнішніми джерелами[8].

2.2 Існуючі рішення у веб-технології та їх призначення

Величезна кількість Інтернет-технологій в сучасному світі може змусити розгубитися навіть досвідченого розробника. Однак, існують найбільш використовувані та затребувані інструменти, а також способи їх застосування, які

дозволяють у короткий термін і головне ефективно вирішувати поставлені завдання.

Основою всесвітньої павутини є мова розмітки гіпертексту HTML – Hyper Text Markup Language. Він служить для логічного (змістового) розмітки документа (веб-сторінки)[5: 18].

Найчастіше у web-дизайнера виникає необхідність застосувати у процесі створення html-документа складне форматування – від абзацу до абзацу змінювати шрифт, розташування тексту, його колір, формувати різні таблиці даних[4].

Можна вирішити цю проблему за допомогою стандартних засобів HTML: описувати кожен абзац окремим набором команд, але можна піти іншим шляхом: включити до сторінки опис CSS або підключити зовнішній файл, виконаний у стандарті CSS – Cascading Style Sheets (каскадні таблиці стилів), в якому за допомогою спеціальної макромови один раз жорстко. Іншими словами, файл CSS виконує роль деякого шаблону, що використовується для форматування тексту, таблиць та інших елементів у документі HTML. Є можливість підключати той самий фізичний файл CSS до різних web-сторінок сайту. CSS можна використовувати на будь-якому сервері без будь-яких обмежень, оскільки команди CSS виконуються безпосередньо на комп'ютері користувача. До недоліків цієї технології можна лише віднести відсутність підтримки CSS старими браузером (Internet Explorer і Netscape Navigator нижче 4-ої версії) і трохи різний набір властивостей CSS, що підтримується останніми версіями цих двох браузерів[16: 17].

Для надання веб-сторінкам динамізму (випадають меню, анімація) використовуються мови написання скриптів. Стандартною скриптовою мовою у всесвітньому павутинні є JavaScript. Ядром JavaScript є ECMAScript.

JavaScript - це мова програмування, яка використовується у складі сторінок HTML для збільшення можливостей. Він був розроблений фірмою Netscape на базі мови Sun's Java корпорації Sun. JavaScript є ніби надбудовою стандарту HTML і значно розширює можливості html-документа, створеного з використанням цієї технології. JavaScript інтегрується у файл HTML у вигляді кількох рядків коду

(наприклад, це може бути функція, що викликається на виконання спеціальною командою). Вбудований у браузер інтерпретатор JavaScript сприймає і скрипт, і сам HTML-код як єдиний документ, обробляючи і ті, й інші дані одночасно[2]

Модуль Java на відміну від JavaScript інтегрується в сторінку, що його використовує, тільки після завантаження і виконання самостійної програми (програми) з розширенням .class, такі програми називаються аплетами. Апплет також викликається з html-файлу відповідною командою, але завантажується, ініціалізується та запускається на виконання у вигляді окремої програми, у фоновому режимі, а до виконання аплету на його місці ви можете лише споглядати сірий прямокутник. Підтримка цієї технології здійснюється за допомогою так званої "Віртуальної машини Java".

Аплети Java в основному використовувалися для надання інтерактивності та візуальної краси web-сторінкам. Але оскільки аплет завантажувалися досить повільно (через невеличкого розміру class файлів) і після написання коду необхідно було створити безпосередньо аплет за допомогою спеціального компілятора, а також можливість створювати ці інтерактивні елементи з використанням того ж JavaScript, а також DHTML і CSS, зумовили досить рідкісне застосування технології Java у вигляді аплетів сьогодні.

За допомогою технології Java/JavaScript можна надати своїй сторінці елементи інтерактивності, формувати, компонувати та повністю контролювати формат спливаючих вікон та вбудованих фреймів, організовувати такі активні елементи, як "годинник", "рядки, що біжать" та іншу анімацію, створити чат. Більшість web-камер, що передають на сайт "живе" зображення, також працюють на базі відповідних програм Java.

Використання цих технологій не потребує встановлення та налаштування на сервері будь-яких додаткових модулів, оскільки скрипти та аплет виконуються безпосередньо на комп'ютері користувача. Браузери старих версій (Internet Explorer і Netscape Navigator нижче з 4-ої версії), що не підтримують Java / JavaScript, не зможуть правильно відображати веб-сторінки, створені за допомогою цих технологій. Але зараз таких браузерів лише близько 3-4%.

HTML, CSS, JavaScript - є мовами, за допомогою яких можна створювати складні веб-сайти. Але це лише лінгвістичне забезпечення, тоді як у браузерях документи подаються у вигляді набору об'єктів, безліч типів яких є об'єктною моделлю браузера (BOM). Об'єктна модель браузера є унікальною для кожної моделі і таким чином виникають проблеми при створенні міжбраузерних додатків. Тому веб-консорціум запропонував об'єктну модель документа (DOM), яка є стандартним способом представлення веб-сторінок за допомогою набору об'єктів[4].

На відміну від об'єктної моделі браузера DOM містить набір об'єктів лише для вмісту документа і не має об'єктів, що дозволяють керувати вікнами та рамками вікон. При написанні програм з метою підтримки міжбраузерної переносимості необхідно дотримуватися стандартів DOM, а об'єктної моделі браузера вдаватися лише за крайньої необхідності. Така потреба може виникнути, наприклад, при керуванні вікнами та рядком стану[4].

Сукупність HTML, CSS, JavaScript та DOM часто називають динамічним HTML – Dynamic HTML або DHTML[5: 18].

DHTML (Dynamic Hyper Text Markup Language, динамічна мова розмітки гіпертексту) є розширенням стандарту HTML і дозволяє створювати web-сторінки, що включають такі інтерактивні елементи, як рухомий фон, розташований під статичним вмістом документа, рухомі об'єкти, випадають меню, кнопки, що підсвічуються при наведенні курсору. За великим рахунком DHTML є "середнім арифметичним" між технологіями HTML та JavaScript. Цей стандарт використовує прості сценарії, підготовлені за допомогою макромови, що інтерпретується, оброблюваного браузером спільно з кодом HTML. Такі сценарії називаються "скриплетами".

Для створення скриплетів використовуються стандартні розширення DHTML та будь-яка макромова, що підтримує директиви інтерфейсу ActiveX. DHTML розпізнається браузерами Microsoft Internet Explorer, починаючи з версії 4.0 та вище.

PHP (Personal Home Page tools) - це ще одна мова, що інтерпретується, призначена для надання web-сторінкам елементів інтерактивності. Код, написаний мовою PHP, вбудовується в документ HTML подібно до підпрограми: в ту ділянку документа, де необхідно розмістити інтерактивний елемент, просто вставляється сценарій PHP. Мнемоніка цієї мови базується на синтаксисі PERL, Java і C, завдяки чому не викликає будь-яких труднощів щодо. Методики, які дозволяють серверам коректно розпізнавати файли, що містять скрипти PHP, різні і залежать передусім від типу конкретного сервера. Як правило, достатньо призначити такому файлу розширення `.php`, іноді - з додаванням номера версії мови, наприклад `.php3` або `.php4`.

Технологія PHP дозволяє організовувати на web-сторінці лічильник відвідувань, підраховувати статистику звернень до тих чи інших розділів сайту, захистити доступ до будь-якого html-документа паролем та багато іншого. Серед недоліків PHP слід зазначити, що дана технологія підтримується далеко не всіма серверами Інтернету.

Технологія CGI (Common Gateway Interface) передбачає використання у складі ресурсу Інтернету інтерактивних елементів з урахуванням додатків, щоб забезпечити передачу потоку даних від об'єкта до об'єкта. Саме так організовано у Всесвітній мережі більшість чатів, конференцій (форумів), дощок оголошень, гостьових книг, пошукових машин та рейтингових систем. Спрощено принцип роботи CGI виглядає так: наприклад, користувач заповнює на web-сторінці ту чи іншу форму і натискає на кнопку, після чого інформація з форми передається в CGI-скрипт, який запускається на виконання та обробляє отриману інформацію.

Серед переваг CGI слід відзначити їхню незалежність від клієнтського програмного забезпечення - цю технологію зможе застосовувати кожен користувач, який переглядає вміст сервера за допомогою браузера практично будь-якої версії.

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	▲	Python	8.262%	+2.39%
4	3	▼	C++	8.126%	+1.67%
5	6	▲	Visual Basic .NET	6.429%	+2.34%
6	5	▼	C#	3.267%	-1.80%
7	8	▲	JavaScript	2.426%	-1.49%
8	7	▼	PHP	2.420%	-1.59%
9	10	▲	SQL	1.926%	-0.76%
10	14	▲	Objective-C	1.681%	-0.09%

Рис. 2.1 -Статистика ТІОВЕ щодо використання веб-технологій

Як видно з вищесказаного, всі веб-технології тісно взаємопов'язані. Розуміння цього факту дозволить легше усвідомити призначення того чи іншого механізму, який використовується під час створення веб-додатків. Технології розвиваються та удосконалюються з кожним роком[1].

2.3 Вибір інструменту для розробки веб-орієнтованої системи автосалону

Зважаючи на статистику застосування різних веб-технологій у розробці та, враховуючи їх призначення, а також сильні та слабкі сторони, було прийнято рішення подивитися у бік JavaScript. Однак, відомі ресурси, такі як (вставити щось), сучасні тенденції і власні напрацювання можна прийти до висновку про недоцільність використання класичного JavaScript, а подивитися в бік фреймворків і бібліотек, створених на його основі, так як вони мають більш просунутий і в той же час полегшений функціонал і менше граничний завдань.

Насамперед розглянемо особливості проектування з допомогою фреймворків, як фронтенду так і бекенду, з'ясуємо, у чому полягає основна різниця між ними та виведемо основні принципи, які у кожному з цих інструментів проектування[2].

Фреймворки надають чітку структуру докладання та реалізуються з використанням так званих «патернів проектування». Найбільш широко поширені

такі патерни: MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel).

Переваги побудови програми на JS-фреймворку:

- дозволяє легко реалізувати SPA (Single Page Application);
- підтримує структуру програми
- точний та добре структурований код;
- модульність програми, можливість одночасного використання кількома способами;
- можливість швидко створити мобільний та/або настільний кросплатформовий додаток з веб-версії за допомогою систем типу PhoneGap або Apache Cordova.

Frontend — це те, що відображається, а також взаємодіє з користувачем у частині програми. Головними завданнями фронтенду є створення інтерфейсу, забезпечення взаємодії користувача, створення дизайну, забезпечення доступності програм різних пристроях.

Програмісти, що працюють у фронтенді, використовують мови програмування плюс засоби верстки сторінок — HTML, CSS, JavaScript. Для прискорення процесу написання коду використовуються фреймворки React, Angular, Vue.js, також багато інших[1: 27].

1. React (більше 110 тисяч зірок на GitHub) - це ефективна та гнучка декларативна бібліотека JavaScript для складання UI від команди Facebook. Вона дозволяє без зусиль створювати інтерактивний інтерфейс користувача.
2. Vue.js – ще один фреймворк для збирання інтерфейсів користувача. Він включає доступну кореневу бібліотеку, яка в першу чергу вирішує завдання рівня представлення, та екосистему додаткових бібліотек, що дозволяє створювати складні та об'ємні односторінкові програми (Single-Page Applications).
3. Angular - фреймворк від компанії Google, який отримав майже 44 тисячі зірок на GitHub. Являє собою платформу, що спрощує складання додатків у Інтернет. У Angular поєднуються декларативні шаблони, впровадження

залежності, двостороннє зв'язування даних та найкращі практики вирішення проблем розробки. Ця платформа дозволяє збирати програми для веб, мобільних пристроїв та настільних ПК.

Серверна частина, тобто backend, є невід'ємною частиною візуального оформлення, але працює на задньому плані у фоновому режимі. Завданням цієї служби є обробка, зберігання запитів, а також забезпечення безпеки даних, виконання бізнес-логіки програми[1; 2; 27; 6].

При створенні backend-програм використовуються мови програмування Python, Ruby, Java, PHP, .NET та JavaScript фреймворки, такі як node.js. Також використовуються серверні технології, мережеві бази даних, віддалені сховища.

Таблиця 2.1

Фреймворки для backend-програм.

Технологія	Тип	Основне використання	Фреймворк	Плюси	Мінуси
Python	Мова	Веб, наука про дані, автоматизація, ML	Django, Flask	Легка, велика спільнота, популярна в AI/ML	Повільніша за Java/.NET
Ruby	Мова	Веб-розробка	Ruby on Rails	Простий синтаксис, швидка розробка	Менша популярність, не дуже гнучкий для інших задач
Java	Мова	Ентерпрайз, Android, сервери	Spring, Hibernate	Надійна, масштабована, стабільна	Об'ємний код, складніша в навчанні
PHP	Мова	Веб-розробка	Laravel, Symfony	Простий старт, багато CMS (WordPress)	Негативна репутація, застарілий синтаксис (у старих проєктах)
.NET	Платформа	Веб, десктоп, мобільні застосунки	ASP.NET Core, Blazor	Висока продуктивність, підтримка Microsoft, C#	Залежність від екосистеми Microsoft (частково)
Node.js	Рантайм (JS)	Серверний JS, API, реального часу застосунки	Express, NestJS	Асинхронність, одна мова для фронту і беку	Callback hell (якщо не використовувати async/await), трохи нижча продуктивність у CPU-важких

					задачах
--	--	--	--	--	---------

Програмісти backend фокусуються на серверній логіці, базах даних, архітектурі, управлінні API, забезпечуючи стабільну роботу написаних програм.

Взаємодія фронт/бек

Frontend та backend взаємодіють за допомогою API (Application Programming Interface), які дозволяють передавати дані між клієнтом та сервером. Бекенд обробляє запити, що надходять від фронтенду, повертає необхідні дані, які потім відображаються користувачеві[2; 27; 6; 9].

Хорошим прикладом такої взаємодії може бути онлайн-магазин, де фронт відповідає за відображення товарів, інтерфейс кошика, а бек обробляє інформацію про товари, замовлення, виконує операції з базою даних.

Різниця фронтенду та бекенду

Відмінності у завданнях та відповідальності

Інтерфейс і серверна частина є частинами одного процесу, що взаємодоповнюють, але існують відмінності в завданнях, у відповідальності фахівців. Автор інтерфейсу фокусується на тому, що бачить користувач: макет, дизайн, анімація. Його відповідальність полягає у створенні інтуїтивно зрозумілого, гарного інтерфейсу[27].

Серверна частина обробляє "невидиму" частину програми. Програміст відповідає за серверну логіку, базу даних, управління API, безпеку, виконання серверних скриптів. Він гарантує, що всі дані обробляються, зберігаються правильно, а програма працює стабільно, безпечно без збоїв.

Приклади та кейс-стаді для наочності

Прикладом, що ілюструє різницю між backend та frontend, може бути процес створення сайту. Фронтенд-програміст створює дизайн сторінки, має в своєму розпорядженні елементи управління - кнопки, форми введення. Бекенд-програміст забезпечує коректну обробку даних, що вводяться користувачем у форми, а також щоб коректно працювали всі необхідні серверні взаємодії[27].

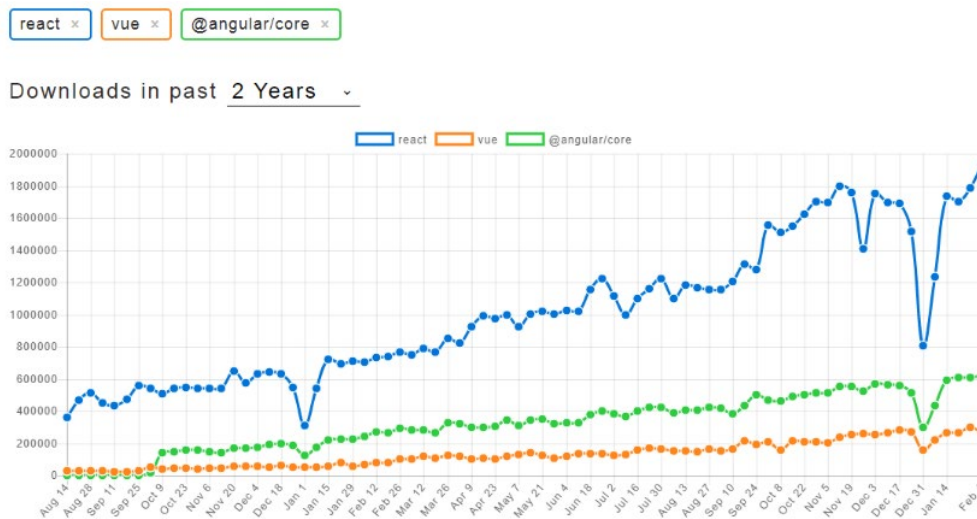


Рис. 2.2 –Статистика завантажень 3 фреймворків за даними GitHub

1.1.1 React.js та його основні складові

React — це інструмент для створення інтерфейсів користувача. Його головне завдання — забезпечення виведення на екран того, що можна побачити на веб-сторінках. React значно полегшує створення інтерфейсів завдяки розбиттю кожної сторінки на невеликі фрагменти компонентами. Кожен компонент — це JavaScript-функція, яка повертає ділянку коду, який представляє фрагмент сторінки. Здійснюється у певному порядку виклик компонентів та збираються разом отримані результати — так формується сторінка, яку бачить користувач. Для оптимізації швидкодії компоненти React спочатку перетворюються на керовану модель Virtual DOM, а потім за допомогою спеціальних інструментів написаний на JSX код на зрозумілий браузеру JavaScript[1; 2; 9; 27].

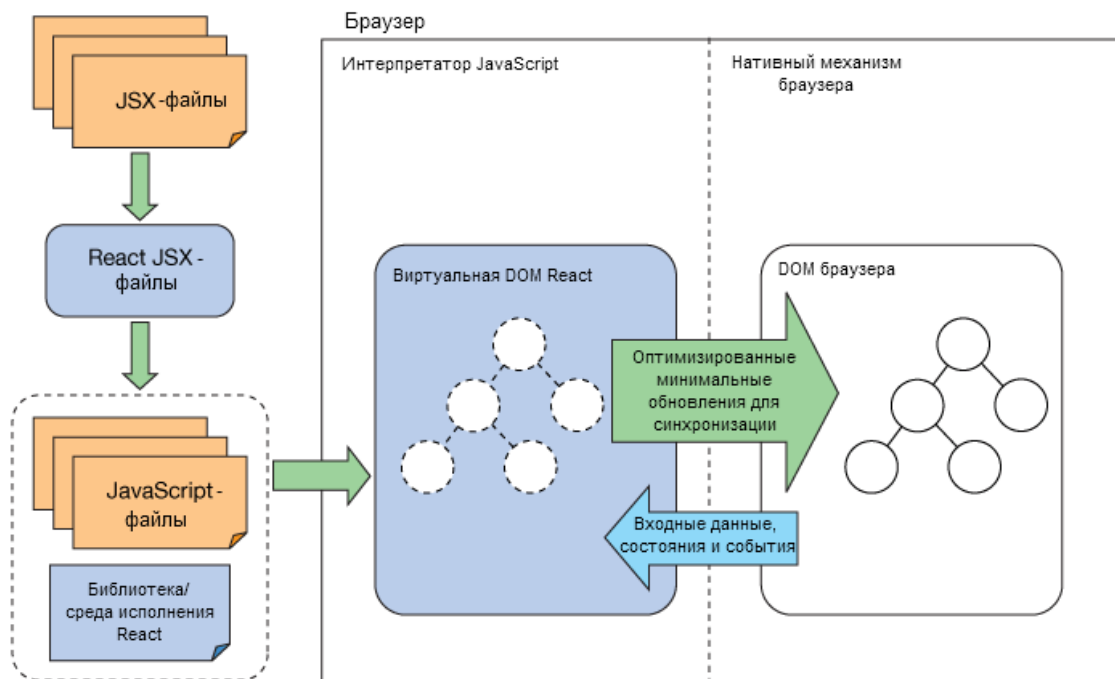


Рис. 2.3 – Механізм роботи бібліотеки React.js

Взагалі, щоб використовувати React необхідно знати та освоїти наступне:

- компоненти React.
- Рендеринг ReactDOM.
- Класи компонентів та функціональні компоненти.
- JSX.
- Стан (state).
- Опрацювання подій.
- Асинхронний метод setState.
- Властивості (props).
- Посилання (refs) [7].

Ось наочний приклад React-компонента.

```
function OurFirstComponent() {
  return (
    <h1>Hello, I am a React Component!</h1>
  );
}
```

Для того, щоб відобразити (відрендерити) компонент на сторінці вдалося до допомоги ReactDOM.

```
const placeWeWantToPutComponent = document.getElementById('hook');
```



```
ReactDOM.render(OurFirstComponent(), placeWeWantToPutComponent);
```

Функція `.getElementById('hook')` використовується для прив'язки рендеринг компонента до певної ділянки html-коду. Компоненти можуть бути інкапсульовані інші компоненти і т.д. Щоб викликати весь ланцюжок у ReactDOM, потрібно вказати назву верхнього компонента. Такий підхід називається композицією. Однак компоненти можна писати і інакше, у вигляді класів JavaScript - класи компонентів.

Продовжимо огляд бібліотеки, щоб зрозуміти всі механізми проектування та побудови додатків на React, а також побачити всі можливості для їх здійснення.

Стан

Стан — це інструмент, що дозволяє оновлювати інтерфейс користувача, ґрунтуючись на подіях. Іншими словами — це спеціальний об'єкт усередині компонента.

```
class Container extends React.Component{
  constructor(props) {
    super(props);
    this.state = {isMusicPlaying: false};
  }
```

Оновити (змінити) стан можна за допомогою єдиної функції `this.setState()`. Після її викликом слід перемальовування всього компонент, якого воно належить. Це одна з головних особливих бібліотек React.

```
handleClick(){
  if(this.state.isMusicPlayig){
    this.setState({isMusicPlaying: false});
  }
  else {
    this.setState({isMusicPlaying: true});
  }
};
```

[15]

Властивості

Під властивостями можна розуміти опції компонента. Вони надаються як аргументи компонента і виглядають так само, як атрибути HTML.

Події

Вони також можуть служити каталізатором для оновлення стану і, як наслідок, перемальовки компонента.

Посилання

Якщо нам потрібно звернутися до певного тегу та викликати його методи. Звичайно, можна використовувати традиційний JavaScript-підхід - написати `document.getElementById(тег).метод()`. Хоча краще звернутися до функціоналу React. Ми призначаємо елементу атрибут, який називається `ref`, який приймає функцію.

Ця функція, як перший аргумент, приймає елемент і надає його `this.<елемент>`. Далі до цієї конструкцію через крапку пишеться ім'я функції, яку слід викликати[14].

Таким чином вдалося познайомитися з новою для нас js-бібліотекою і навіть зробити перші кроки в її освоєнні. Простота і гнучкість коду, легкість написання та підтримки зв'язків між компонентами, а також хороша поєднання з популярною No-SQL БД зробили React найкращим інструментом для вирішення поставленого завдання – але про вибір бази даних та його обґрунтування розповідають наступні розділи.

1.1.2 Що таке Node.js і як вона влаштована

Node.js - це середовище виконання JavaScript, побудоване на движку V8 від Google, яке:

Компілює та виконує JavaScript-код, перетворюючи його на машинний код, зрозумілий для процесора. Завдяки цьому JavaScript працює дуже швидко та ефективно.

Не тільки використовує двигун V8, але й розширює його функціональність, додаючи різні модулі та API для роботи з файлами, мережею, базами даних та іншими ресурсами.

Надає подієвий цикл (event loop) – механізм, який дозволяє обробляти асинхронні операції без блокування основного потоку виконання. Подієвий цикл реєструє функції-обробники (callback) для різних подій (наприклад, завершення

читання файлу або отримання відповіді сервера) і викликає їх по черзі, коли події відбуваються.

Переваги node.js перед Python та PHP

- Єдина мова: полегшує розробку та підтримку коду. Fullstack-розробнику не потрібно вивчати та перемикатися між різними мовами та середовищами розробки.

- JavaScript-фреймворки: пропонують інструменти та функції для обробки HTTP-запитів, маршрутизації, обробки даних та інших завдань, пов'язаних із backend-розробкою.

- Асинхронне програмування: дозволяє ефективно обробляти безліч запитів за допомогою асинхронної моделі виконання коду. Це особливо корисно для розробки високопродуктивних програм, таких як чати, стрімінгові сервіси, де потрібна одночасна обробка великої кількості запитів.

- Інтеграція з фронтендом: ви можете використовувати JSON для представлення даних та легко маніпулювати ними як на клієнтській, так і серверній стороні[6; 11; 23].

Як виглядає код на Node.js

Код на Node.js виглядає так само, як код JavaScript, за винятком того, що він використовує спеціальні модулі і API для роботи з сервером. Давайте створимо простий сервер:

```
// Подключаем модуль http
const http = require('http');

// Создаем сервер
const server = http.createServer((req, res) => {
  // Отправляем ответ в виде простого текста
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello, Node.js!');
});

// Запускаем сервер на порту 3000
server.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Рис. 2.4 –творення простого сервера на node.js

Цей код створює сервер, який слухає порт 3000 та надсилає вітальне повідомлення кожному, хто звертається до нього. Щоб запустити цей код, потрібно зберегти його у файлі з розширенням .js (наприклад, server.js) і виконати команду `node server.js` у терміналі. Після цього можна відкрити браузер та перейти за адресою `http://localhost:3000`. Там має з'явитися повідомлення Hello, Node.js!.

2.4 Вибір типу СУБД

На сьогоднішній день існують два найбільш популярні типи баз даних – реляційні та нереляційні. Давайте розберемося, у чому різниця.

Основні відмінності ховаються в деяких характерних рисах: як вони спроектовані, які типи даних підтримують і як зберігають інформацію.

Реляційні БД зберігають структуровані дані, які зазвичай становлять об'єкти реального світу. Це можуть бути відомості про людину або вміст кошика для товарів у магазині, згруповані в таблицях, формат яких заданий на етапі проектування сховища.

Нереляційні БД улаштовані інакше. Наприклад, документоорієнтовані бази зберігають інформацію як ієрархічних структур даних. Йдеться про об'єкти з довільним набором атрибутів. Те, що у реляційної БД буде розбито кілька взаємозалежних таблиць, в нереляційної може зберігатися як цілісної сутності.

Реляційні бази даних складаються з кількох ключових компонентів:

Таблиці: Основні структури для зберігання даних. Кожна таблиця складається з рядків і стовпців, де кожен рядок представляє окремий запис, а кожен стовпець — атрибут даних.

Первинний ключ: Унікальний ідентифікатор для кожного запису в таблиці. Він гарантує, що кожен запис може бути однозначно ідентифікований.

Зовнішній ключ: Поле в одній таблиці, яке посилається на первинний ключ іншої таблиці, створюючи зв'язок між таблицями.

Індекси: Структури, що підвищують швидкість доступу до даних у таблиці. Індекси дозволяють швидко знаходити рядки за певними критеріями.

Запити: Використовуються для отримання даних з таблиць. Найпоширенішою мовою запитів для реляційних баз даних є SQL (Structured Query Language)[31; 32].



Рис. 2.5 –творення простого сервера на node.js

Таблиця 2.2

Порівняння реляційних та нереляційних баз даних

Критерій	Реляційні БД (SQL)	Нереляційні БД (NoSQL)
1. Структура зберігання	Таблиці з рядками і стовпцями	Документи (JSON, BSON), граfi, колоночні сховища, ключ-значення
2. Схема	Жорстка, чітко визначена. Схема має бути визначена перед зберіганням даних	Гнучка, динамічна. Схема може змінюватися "на ходу"
3. Типи даних	Переважно структуровані дані	Підтримка структурованих, напівструктурованих і неструктурованих даних
4. Запитна мова	SQL (Structured Query Language)	Залежить від СУБД: MongoDB Query Language, CQL (Cassandra), власні API
5. Підтримка транзакцій	Повна (ACID: атомарність,	Часткова (BASE: Basically

	узгодженість, ізоляція, надійність)	Available, Soft state, Eventually consistent)
6. Зв'язки між даними	Використання зовнішніх ключів, JOIN-операцій	Найчастіше — денормалізація, зв'язки описуються вручну або через вкладені об'єкти
7. Масштабування	Вертикальне (додавання потужності одному серверу)	Горизонтальне (додавання нових серверів або вузлів)
8. Продуктивність	Висока при складних запитах і транзакціях	Висока при великих обсягах даних, реальному часу, високій паралельності
9. Надійність при збоях	Висока. Зазвичай має реплікацію та резервне копіювання	Також висока, але залежить від конкретної реалізації
10. Гнучкість зміни структури	Низька: потребує ALTER TABLE та ретельного перепроєктування	Висока: зміна структури на рівні одного документа чи колекції
11. Швидкість розробки	Повільніша через необхідність визначення схеми	Швидша, особливо на ранніх етапах
12. Підтримка великих обсягів	Обмежено (масштабування ускладнене)	Добре працює з Big Data, великими навантаженнями
13. Приклади СУБД	MySQL, PostgreSQL, Oracle, Microsoft SQL Server, MariaDB	MongoDB, Cassandra, Redis, Couchbase, Neo4j, DynamoDB
14. Крайні випадки використання	Банки, фінанси, CRM, ERP, транзакційні системи	Соцмережі, IoT, Big Data, веб-програми, системи реального часу
15. Складність підтримки	Висока (потрібні DBA, оптимізація запитів)	Залежить від типу, але зазвичай простіше для початкового адміністрування
16. Реплікація та шардінг	Реплікація можлива, шардінг складний	Часто вбудована підтримка шардінгу і реплікації
17. Безпека	Високий рівень контролю доступу, аудит	Варіюється — може бути менш контрольований, але часто має вбудовані механізми
18. Зрілість технології	Дуже зріла, багаторічне використання в індустрії	Відносно нові, але активно розвиваються й набирають популярності

Висновок по розділу 2

Було складено концептуальну модель предметної області, проведено її опис. Визначено види міжатабличних зв'язків. Розглянуто основні алгоритми з проектування даталогічної схеми та методи щодо її практичної реалізації.

Проведено характеристику різних реляційних СУБД, обґрунтовано вибір MySQL, завантажена спроектована база даних

– Автомобіля (код авто, довжина, ширина, висота, маса, вартість, кількість дверей, кількість місць для сидіння, об'єм багажника, ємність паливного бака, максимальна швидкість, час розгону до 100 км/год, витрата палива, клімат-контроль, розмір шин, розмір дисків) ;

– Країна (код країни, назва);

– Привід (код приводу, тип приводу);

– Електронні асистенти (код ел. асистенту, назва);

– Двигун(код двигуна, тип, кількість циліндрів, об'єм двигуна, потужність, момент оберту);

– Коробка передач (код к.п, назва);

– Модель (код моделі, назва моделі, зображення, модельний рік, ціна)

– Зображення (код зображення., посилання);

– Підсилювач керма (код підсилювача керма, тип);

– Тест-драйв(код тест-драйву);

– Продажі(код продажі, категорія);

– Покупець(код покупця, ПІБ);

– Менеджер(код менеджера, ПІБ).

Беручи за основу теоретичні висновки, зроблені раніше, вдалося приступити до побудови інфологічної схеми бази даних «Автосалон». Оскільки більшість таблиць є довідниками, то зв'язки в основному будуть представлені відношенням 1 : М, наприклад

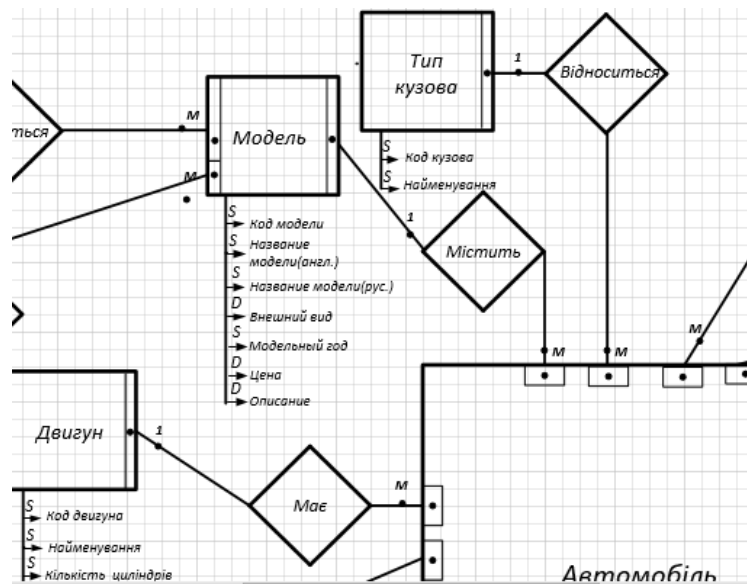


Рисунок 3.2 – приклад відношення 1:M

Відношення M:M також є, між таблицями електронні помічники і властивості автомобіля.

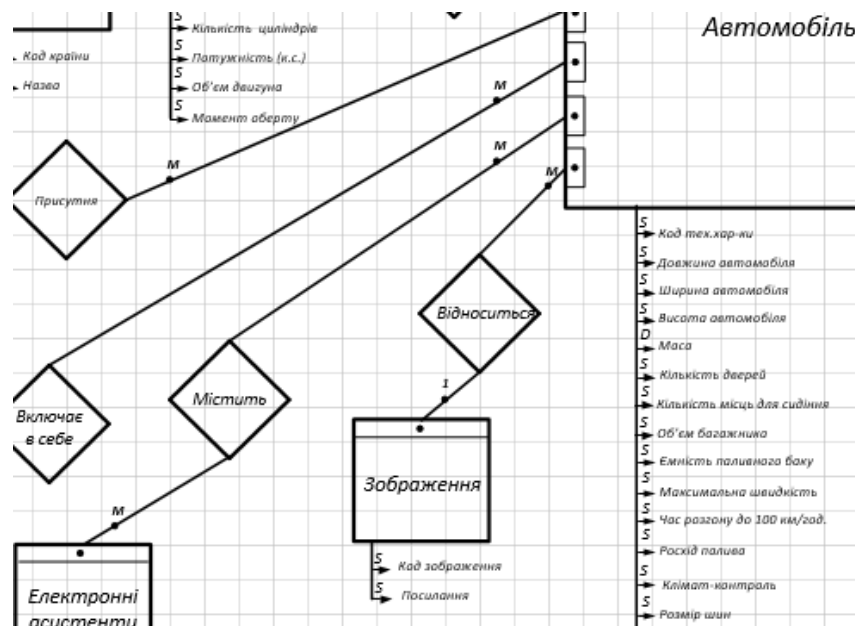


Рис. 3.3 – приклад відношення M:M

Атрибути здебільшого прості, статичні та динамічні, зв'язки мають обов'язковий клас власності[8; 10; 25].

Ось підсумкова таблиця зв'язків бази даних:

Зв'язок між таблицями бази даних «Автосалон»

Таблиця 1	Таблиця 2	Вид зв'язку
Підвіска	Автомобіль	1: М
Тип кузова	Автомобіль	1: М
Марка	Модель	1: М
Країна	Модель	1:М
Привід	Автомобіль	1: М
Електронні асистенти	Автомобіль	М: М
Двигун	Автомобіль	1: М
Коробка передач	Автомобіль	1: М
Зображення	Модель	М:1
Підсилювач керма	Автомобіль	1: М
Модель	Автомобіль	1: М
Автомобіль	Тест-драйв	1:М
Менеджер	Продажі	1:М
Покупець	Продажі	1:М
Покупець	Тест-драйв	1:М
Менеджер	Тест-драйв	1:М

3.2 Побудова даталогічної схеми бази даних

Згідно з побудованою інфологічною моделлю ми отримали 18 таблиць, у тому числі зі зв'язком М:М. Тому необхідно додавання до бази ще однієї таблиці, яка міститиме первинні ключі обох таблиць, тобто замінимо цей зв'язок на 2 зв'язку 1:М[8; 10; 25].

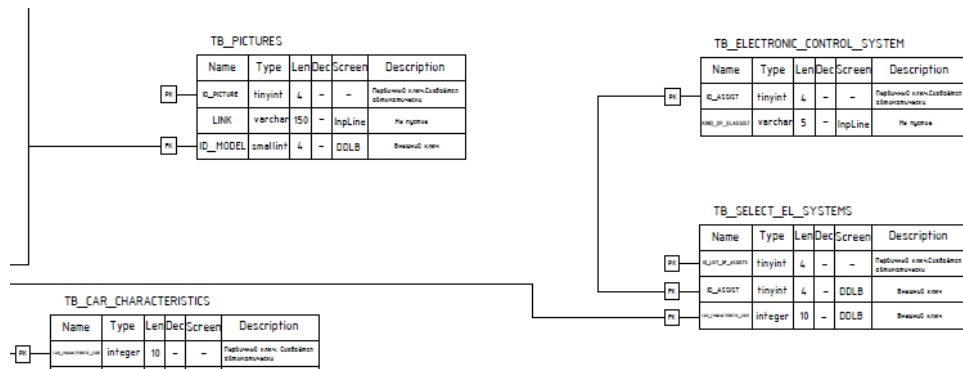


Рис. 3.4 – приклад відношення М:М

Якщо дві таблиці пов'язані ставленням 1:М, то первинний ключ таблиці 1 поміщається в таблицю М як зовнішній.

- suspender(id_suspender, kind_of_suspender)
- carcase_type(id_carcase, type_of_carcase)
- car_brend(id_brend, brend_name, logo)
- car(car_code, id_carcase, id_model, car_length, car_width, car_height, weight, count_of_doors, count_of_seats, boot_volume, id_drunit, id_engine, fuel_tank_capacity, max_speed, acceleration_time, fuel_consumption, id_gearbox, id_suspender, id_power_steering, climate_control, wheel_size, disc_size);
- chassis_suspender(id_chassis_suspender, kind_of_cs_suspender)
- country(id_country, c_name)
- drive_unit(id_drunit, dr_unit)
- electronic_control_systems(id_assist, kind_of_lassist)
- engine(id_engine, type_of_engine, count_of_cylinders, engine_power, engine_capacity, engine_moment)
- gearbox(id_gearbox, type_of_gearbox)
- model(id_model, model_name, id_brend, main_view_link, id_country, model_year, price)
- pictures(id_picture, link, id_model)
- power_steering(id_power_steering, kind_of_powsteering)
- select_el_systems(id_list_of_assist, id_assist, car_code)
- test_drive(test_drive_code, date, time, id_manager, id_customer, id_car);

3.3 Характеристики різних СУБД SQL.

Відомі системи управління реляційними базами даних

Існує кілька популярних систем управління реляційними базами даних (RDBMS), кожна з яких має свої особливості та переваги:

1. MySQL

Загальний опис:

- Реляційна СУБД з відкритим кодом.
- Розроблена спочатку компанією MySQL AB, зараз належить Oracle.
- Часто використовується в LAMP-стеку (Linux, Apache, MySQL, PHP/Python).

Переваги:

- Широко підтримується хостингами.
- Добре підходить для веб-застосунків.
- Швидка обробка SELECT-запитів.
- Простота у використанні та налаштуванні.
- Підтримує реплікацію та кластеризацію.

Недоліки:

- Обмежена підтримка складної аналітики порівняно з деякими іншими СУБД.
- Тривалий час Oracle тримає деякі можливості у платній версії[12; 28; 30].

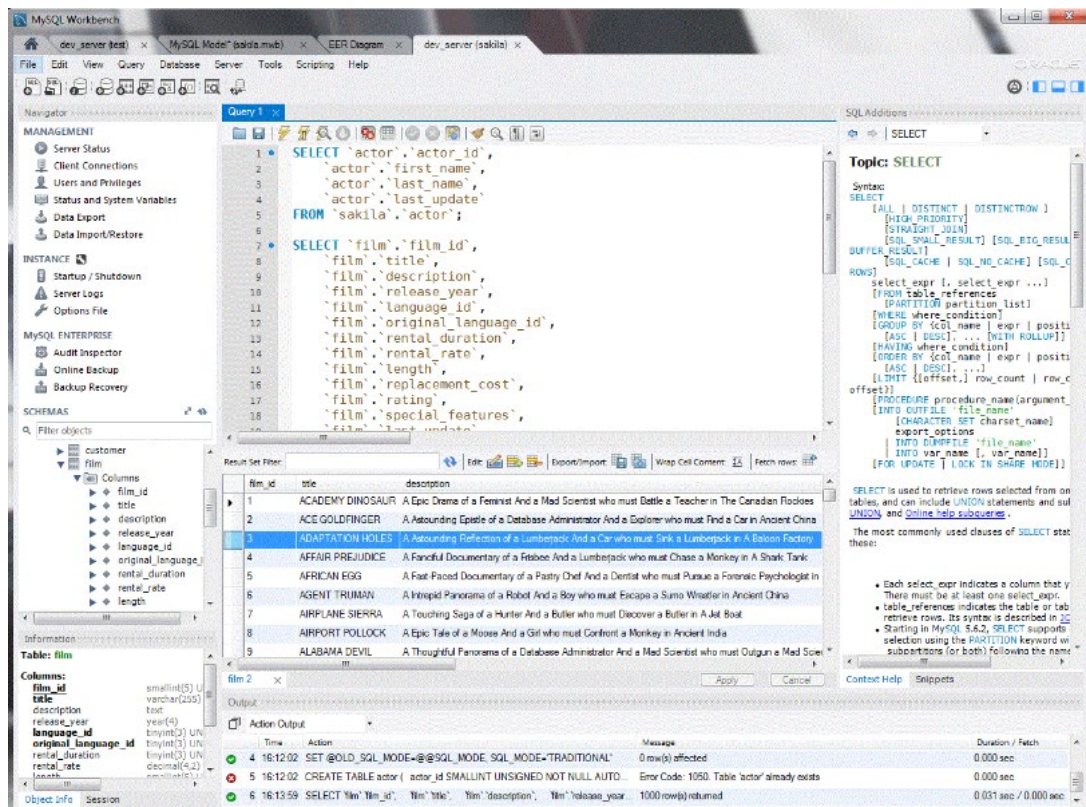


Рис. 3.6 – Приклад інтерфейсу MySQL

2. PostgreSQL

Загальний опис:

– Реляційна СУБД з відкритим кодом, орієнтована на відповідність стандартам SQL.

– Відомий як "найбільш просунутий об'єктно-реляційний СУБД".

Переваги:

- Підтримка складних запитів, CTE, window-функцій.
- Повноцінна транзакційність (ACID).
- Підтримка JSON, XML, розширень (наприклад, PostGIS для геоданих).
- Вища відповідність стандартам SQL.

Недоліки:

- Дещо складніше у налаштуванні для новачків.
- Може бути менш швидким для простих SELECT-запитів порівняно з MySQL.

Коли краще PostgreSQL?

- Якщо важлива складна аналітика, розширення, відповідність стандартам.

– Системи з великими транзакціями, або потребою у зберіганні напівструктурованих даних (JSON) [28; 29]

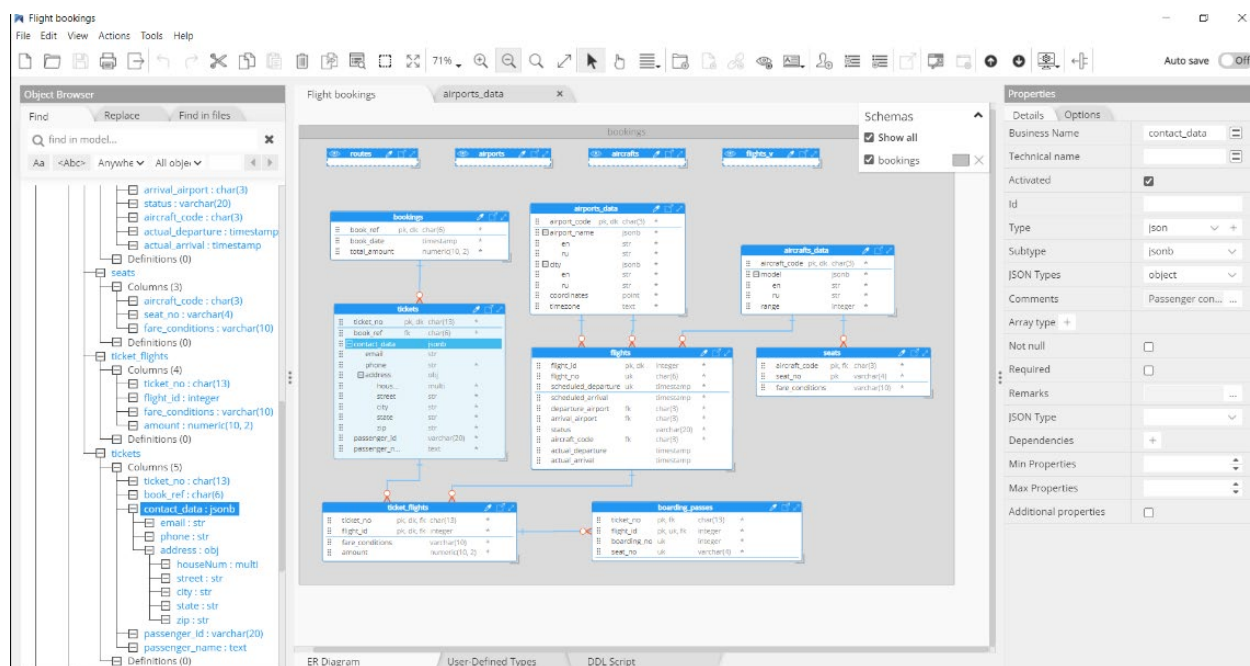


Рис. 3.7 – Приклад інтерфейсу PostgreSQL

3. MariaDB

Загальний опис:

– Форк MySQL, створений творцями оригінального MySQL після покупки Oracle.

– Повністю сумісна з MySQL, але з відкритим управлінням спільнотою.

Переваги:

– Більш відкритий розвиток (community-driven).

– Часто швидше впроваджує нові функції.

– Підтримка альтернативних движків зберігання (Aria, ColumnStore).

– Повністю сумісна з MySQL (у більшості випадків).

Недоліки:

– Деякі зміни можуть порушити повну сумісність з MySQL у довгостроковій перспективі.

– Менше поширена, ніж MySQL.

Коли краще MariaDB?

– Коли потрібна максимальна сумісність з MySQL, але без контролю Oracle.

– Коли важлива відкритість розробки.

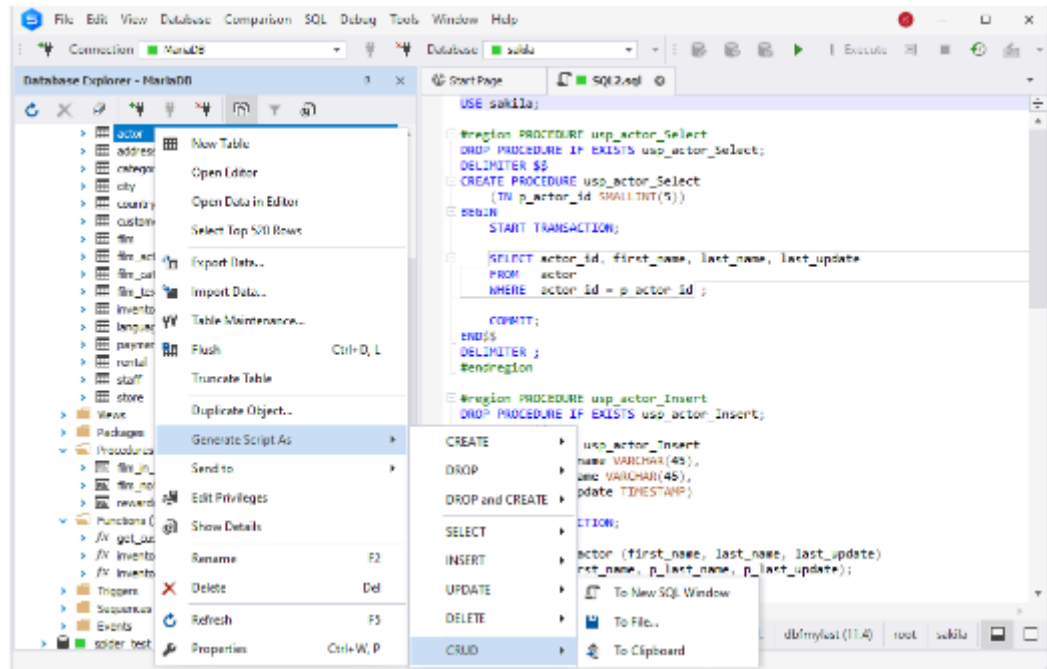


Рис. 3.8 – Приклад інтерфейсу MariaDB

4. SQLite

Загальний опис:

- Надлегка реляційна СУБД, яка зберігає всю базу в одному файлі.
- Не вимагає сервера, дуже проста у використанні.

Переваги:

- Ідеально для мобільних застосунків, вбудованих систем.
- Не потребує інсталяції або адміністрування сервера.
- Дуже швидка для невеликих обсягів даних.

Недоліки:

- Не підходить для багатокористувацьких середовищ або великих систем.
- Обмежена функціональність у порівнянні з серверними СУБД.

Коли краще SQLite?

- Для локальних застосунків, мобільних аплікацій, вбудованих систем.
- Коли потрібна проста, файлова БД без серверної частини.

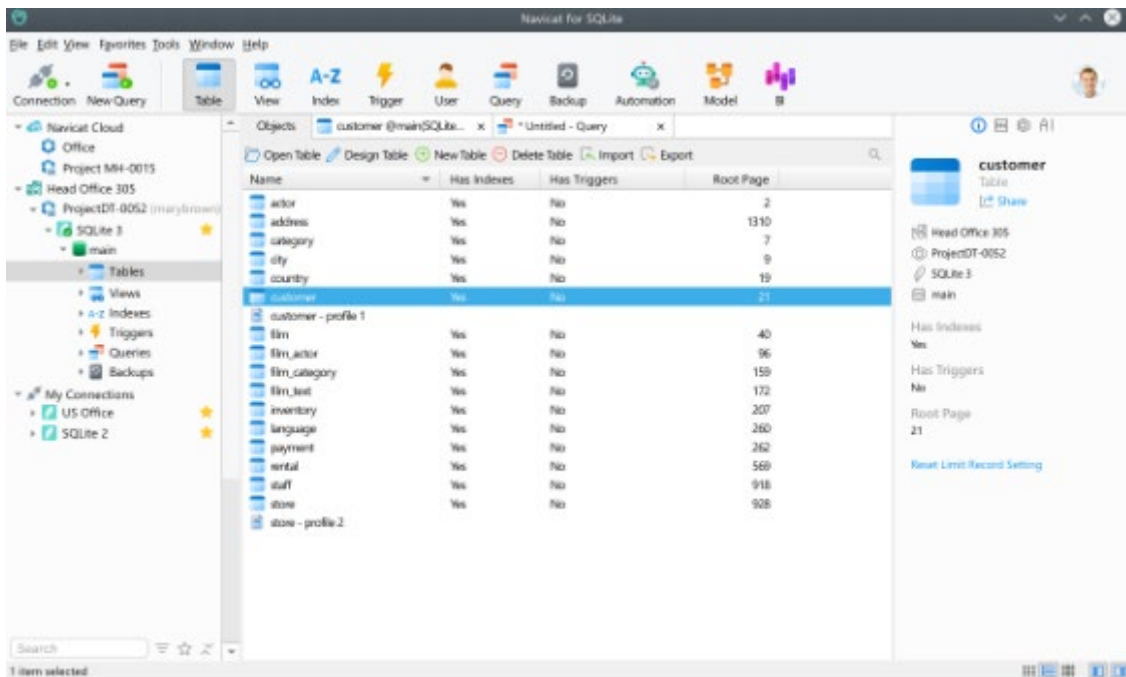


Рис. 3.9 – Приклад інтерфейсу SQLite

5. Microsoft SQL Server

Загальний опис:

- Комерційна СУБД від Microsoft.
- Добре інтегрується з Windows-інфраструктурою та продуктами Microsoft.

Переваги:

- Потужні аналітичні можливості.
- Хороша інтеграція з Power BI, .NET, Excel.
- Зручне адміністрування через SQL Server Management Studio.

Недоліки:

- Платна (є безкоштовна версія Express з обмеженнями).
- Переважно для Windows-середовищ.

Коли краще SQL Server?

- Для підприємств, які працюють в екосистемі Microsoft.
- Коли потрібна глибока інтеграція з бізнес-аналітикою (BI).[30]

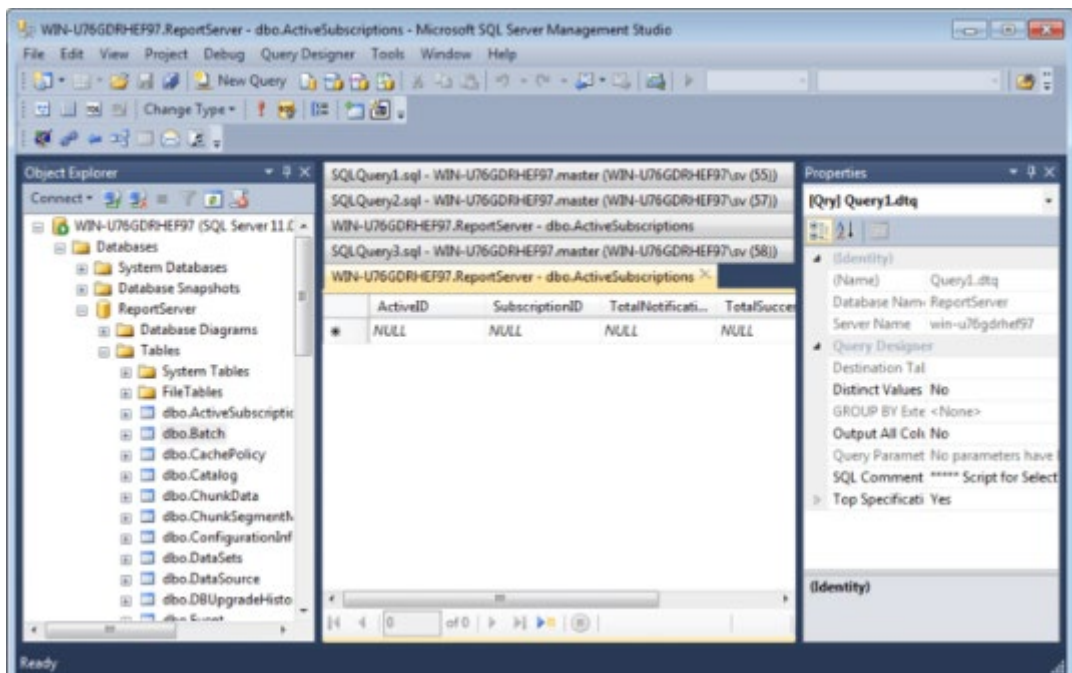


Рис. 3.10 – Приклад інтерфейсу Microsoft SQL Server

6. Oracle Database

Загальний опис:

– Потужна комерційна СУБД, що використовується у великих підприємствах.

– Висока масштабованість, підтримка складних транзакцій.

Переваги:

– Надійність, масштабованість, потужна реплікація.

– Гнучкість у проєктуванні та адмініструванні.

Недоліки:

– Дуже дорога.

– Складне адміністрування, високий поріг входу.

Коли краще Oracle?

– У великих корпоративних системах з великим обсягом транзакцій.

– У системах з високими вимогами до надійності та безпеки.

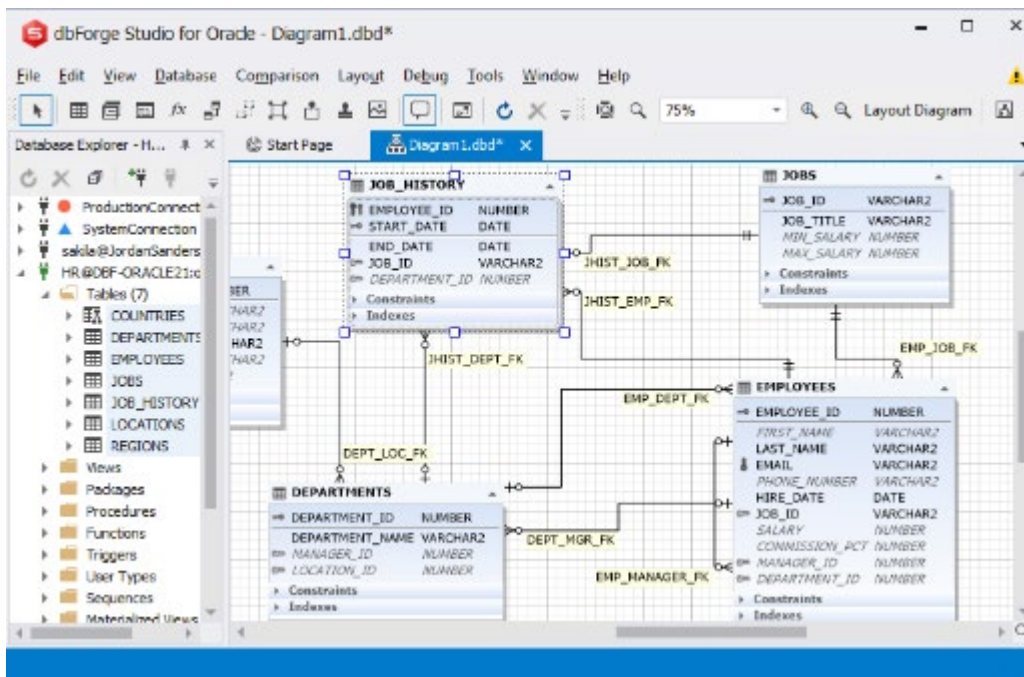


Рис. 3.11 – Приклад інтерфейсу Oracle Database

Зупинимо свій вибір на MySQL, оскільки це доволі зрозуміла та розповсюджена СУБД з відкритим кодом, до якої є багато документації, як офіційної, так і користувацької, до того ж вона сумісна з переважною більшістю технологій розробки програмних продуктів.

Висновок по розділу 3

Було складено концептуальну модель предметної області, проведено її опис. Визначено види міжтабличних зв'язків. Розглянуто основні алгоритми з проектування датовісної схеми та методи щодо її практичної реалізації.

Проведено характеристику різних реляційних СУБД, обгрунтовано вибір MySQL, завантажена спроектована база даних

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГРАМНОГО КОМПЛЕКСУ

4.1 Підготовка до реалізації, налаштування необхідних утиліт та компонентів

Зараз нам слід розглянути, як встановити та налаштувати головний інструмент для реалізації проекту – бібліотеку React.js, оскільки набагато зручніше працювати з ним локально[9; 26;].

Для початку потрібно перевірити, чи є на робочій машині node.js командою `node -v`. оновлення різних компонентів програмного забезпечення Системи управління пакетами активно використовуються в різних дистрибутивах операційної системи Linux та інших Unix-подібних операційних системах. Іншими словами – це інструмент `cmd`, що дозволяє взаємодіяти з браузерами, встановлювати або видаляти компоненти проектів зрозумілий браузеру JavaScript. Зробити це можна за допомогою утиліти `create-react-app` - вона створить характерну структуру папок і файлів, скачає всі необхідні утиліти і налаштує їх таким чином, що можна буде відразу приступати до написання коду. встановлену утиліту, додавши в кінці назву програми. Після завершення процесу створення програми слід перевірити його працездатність[9; 14; 26].

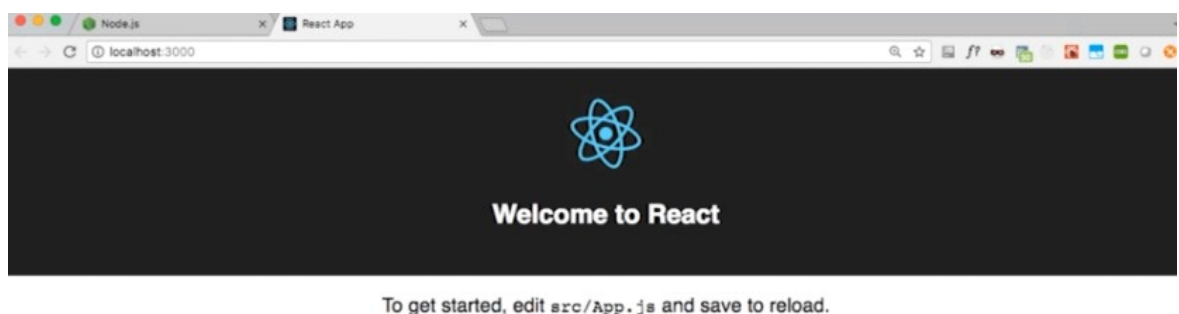


Рис. 4.1 – Стартове вікно react-програми

Таким же чином налаштуємо й фреймворк node.js.

Усе працює, отже можна його можна відкривати в IDE і розпочинати розробку власного програмного продукту.

4.2 Завантаження бази даних «Автосалон» у MySQL

Відкриємо MySQL Workbench та натиснемо «Create a new schema» та додамо наш файл «autohouse.sql». Він має наступний вигляд:

```
10
11
12  -- База даних: `autohouse`
13  --
14
15  -----
16  CREATE TABLE IF NOT EXISTS `carcase_type` (
17    `id_carcase` tinyint(4) NOT NULL AUTO_INCREMENT,
18    `type_of_carcase` varchar(15) NOT NULL,
19    PRIMARY KEY (`id_carcase`)
20  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=13 ;
21
22
23
24  INSERT INTO `carcase_type` (`id_carcase`, `type_of_carcase`) VALUES
25    (1, 'Седан'),
26    (2, 'Універсал'),
27    (3, 'Хетчбек'),
28    (4, 'Кросовер'),
29    (5, 'Позашляховик'),
30    (6, 'Мінівен');
31
```

```

CREATE TABLE IF NOT EXISTS `car` (
  `car_code` int(10) NOT NULL AUTO_INCREMENT,
  `id_carcase` tinyint(4) NOT NULL,
  `id_model` int(10) NOT NULL,
  `car_length` smallint(4) NOT NULL,
  `car_width` smallint(4) NOT NULL,
  `car_height` smallint(4) NOT NULL,
  `weight` smallint(4) NOT NULL,
  `count_of_doors` tinyint(1) NOT NULL,
  `count_of_seats` tinyint(1) NOT NULL,
  `boot_volume` smallint(4) NOT NULL,
  `id_drunit` tinyint(4) NOT NULL,
  `id_engine` tinyint(4) NOT NULL,
  `fuel_tank_capacity` smallint(3) NOT NULL,
  `max_speed` smallint(3) NOT NULL,
  `acceleration_time` decimal(10,1) NOT NULL,
  `fuel_consumption` decimal(10,1) NOT NULL,
  `id_gearbox` tinyint(4) NOT NULL,
  `id_suspender` tinyint(4) NOT NULL,
  `id_power_steering` tinyint(4) NOT NULL,
  `climate_control` enum('Tak','Hi') NOT NULL DEFAULT 'Tak',
  `wheel_size` varchar(10) NOT NULL,
  `disc_size` varchar(10) NOT NULL,
  PRIMARY KEY (`car_code`),
  KEY `id_carcase` (`id_carcase`),
  KEY `id_drunit` (`id_drunit`),
  KEY `id_engine` (`id_engine`),
  KEY `id_gearbox` (`id_gearbox`),
  KEY `id_power_steering` (`id_power_steering`),
  KEY `id_model` (`id_model`),
  KEY `id_suspender` (`id_suspender`),
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;

```

Рис. 4.2 – Фрагмент коду з файлу «autohouse.sql»

У результаті отримаємо наступне[10; 12; 21; 22; 25]:

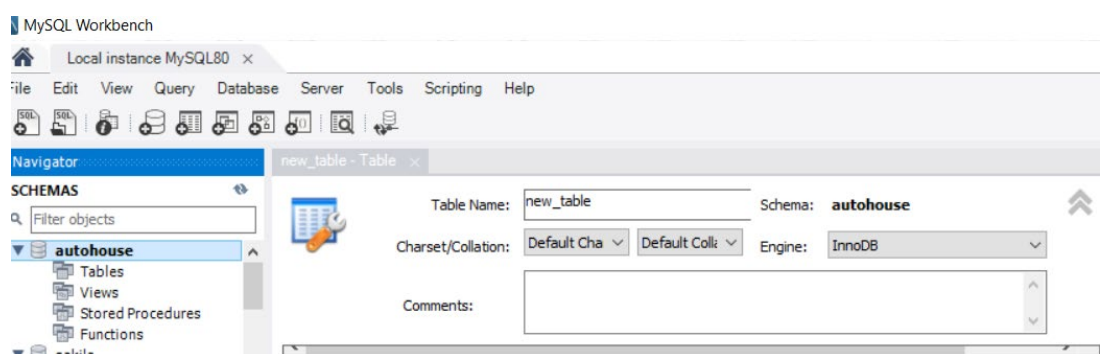


Рис. 4.3 Завантажена база «autohouse.sql»

4.3 Проектування та розробка сайту

Розробка веб-орієнтованої системи з підтримки діяльності автосалону починається з написання сайту. Сайт повинен мати зрозумілий дизайн зі зручним інтерфейсом, бути адаптованим під різні розміри екрану й, звісно, нести всю необхідну інформацію про компанію та асортимент її товару – автомобілі.

Отже, знайшовши у вільному доступі figma-макет, почали написання сайту, що являє собою один із найважливіших компонентів в системі, що розробляється, поряд із БД та панеллю адміністратора.

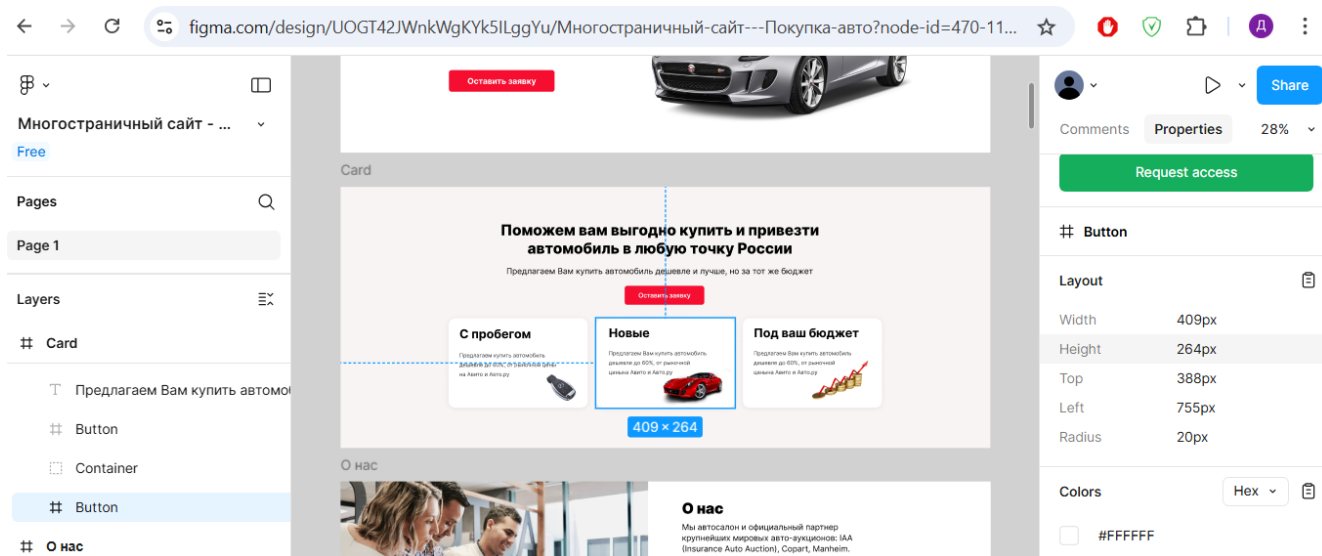


Рис. 4.4 – Фрагмент макету сайту у figma

Для кожної html-сторінки спільними є дві частини, header та footer, перша містить у собі логотип, навігацію по сайту та деяку контактну інформацію, як-от посилання на сторінки в соціальних мережах, друга, у свою чергу, повні контактні дані, час роботи, адресу розташування з міткою на google-мапі, footer і header адаптовані за допомогою технології flex та медіа-запитів, як і увесь сайт[15; 16; 17; 19; 20].



Наші контакти

Адреса

91285, Демидівка, вул. Шевченка 9/10

Час Роботи

Щоденно, з 10:00 до 19:00

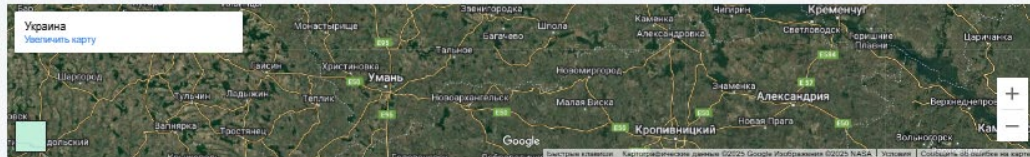
Телефон

+380 (095) 140-20-67
+380 (068) 550-40-57

E-Mail

carmark21@gmail.com

Соціальні Мережі



[Рекавити](#)

[Політика конфіденційності](#)

Наші контакти

Адреса

91285, Демидівка, вул. Шевченка 9/10

Час Роботи

Щоденно, з 10:00 до 19:00

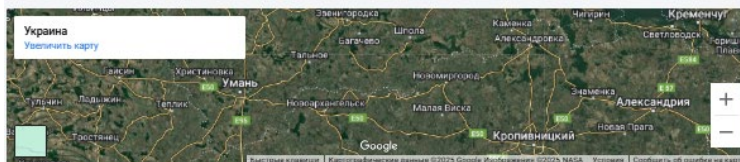
Телефон

+380 (095) 140-20-67
+380 (068) 550-40-57

E-Mail

carmark21@gmail.com

Соціальні Мережі



[Рекавити](#)

[Політика конфіденційності](#)

Рис. 4.5 – Вигляд header і footer (у тому числі й адаптований)

```
.d-flex {  
  display: flex;  
}  
  
.flex-wrap {  
  flex-wrap: wrap;  
}
```



Рис. 4.6 – Заходи з адаптації header та footer (частково притаманні до всього сайту)

Основна сторінка index.html містить у собі окрім тегів, зазначених вище, ще й теги section з інформацією про наявні акційні пропозиції, короткий опис про діяльність компанії, переваги співробітництва саме з цим автосалоном, приклади відгуків задоволених клієнтів. Кожен розділ включає в себе кнопку «Дізнатися

більше», що переносить вас на відповідну тематичну сторінку сайту, що являє собою, по суті, альтернативну навігацію[15; 16; 17; 19; 20].



Про Нас

Каталог

Переваги

Відгуки

Контакти



+380 (095) 140-20-87
+380 (068) 550-40-57

Придбайте TOYOTA CAMRY дешевше на 25%*

*Пропозиція діє до 30.06.2025. Кількість автомобілів обмежена



Дізнатися більше

Ми продаємо японські авто

*Пропозиція діє до 30.06.2025. Кількість автомобілів обмежена

Дізнатися більше

ЗРУЧНО

Lorem ipsum dolor sit



ШВИДКО

Lorem ipsum dolor sit,



ВИГІДНО

Lorem ipsum dolor sit



Про нас

Мы автосалон та офіційний дилер Toyota, mitsubishi, Nissan, Honda.

🚗 Офіційний дилер

✅ Професійний сервіс

📅 На ринку 10 років

Мы берем на себя полное сопровождение сделки до передачи автомобиля в руки клиенту.

Оставить заявку

Клієнти про нас



СЕРГІЙ ДРАЧ

Lorem ipsum dolor sit amet
consectetur, adipisicing elit.
Ratione sed, consequatur illo,
dignissimos placeat minima amet
deserunt molestiae facere culpa
neque facilis doloribus cum
voluptatibus laboriosam ad fugit
quod maiores.

[Докладніше](#)



ОКСАНА ХВИЛЯ

Lorem ipsum dolor sit amet
consectetur, adipisicing elit.
Ratione sed, consequatur illo,
dignissimos placeat minima amet
deserunt molestiae facere culpa
neque facilis doloribus cum
voluptatibus laboriosam ad fugit
quod maiores.

[Докладніше](#)



АНДРІЙ МИРОНЕНКО

Lorem ipsum dolor sit amet
consectetur, adipisicing elit.
Ratione sed, consequatur illo,
dignissimos placeat minima amet
deserunt molestiae facere culpa
neque facilis doloribus cum
voluptatibus laboriosam ad fugit
quod maiores.

[Докладніше](#)

Наші контакти

Адреса

91285, Демидівка, вул. Шевченка 9/10

Час Роботи

Щоденно, з 10:00 до 19:00

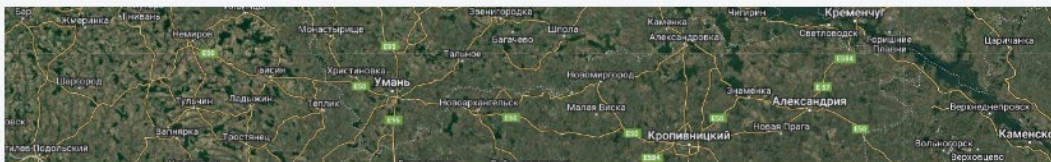
Телефон

+380 (095) 140-20-67
+380 (068) 550-40-57

E-Mail

carmark21@gmail.com

Соціальні Мережі



[Реквізити](#)

[Політика конфіденційності](#)

Рис. 4.7 – Сторінка contacts.html

На сторінці contacts.html маємо форму для зворотнього зв'язку чи відправки коментаря. В останньому випадку свою електронну поштову адресу вказувати не потрібно.



CONTACTS US

Заповніть поля, щоб зв'язатися з нами з усіх цікавлячих Вас питань та пропозицій

Ім'я	Email
------	-------

Ваше звернення

Відправити

Рис. 4.8 – Сторінка contacts.html

Центральними сторінками у розробці сайту є catalog.html та пов'язана з нею model.html. На catalog.html представлені карточки автомобілів обраної марки (перша стоїть за замовчуванням) з відповідними зображеннями та коротким описом технічних характеристик. При натисканні на зображення чи кнопку «Дізнатися більше» відбувається перехід на model.html, де вже містяться більш розширений список технічних характеристик та детальних зображень, реалізованих за допомогою каруселі.

ЯПОНСЬКІ МАРКИ, АВТО ЯКИХ ВИ МОЖЕТЕ ПРИДБАТИ У НАС



TOYOTA



RAV4

Рік випуску: 2024,
Країна: Японія



RAV4

Рік випуску: 2024,
Країна: Японія



RAV4

Рік випуску: 2024,
Країна: Японія

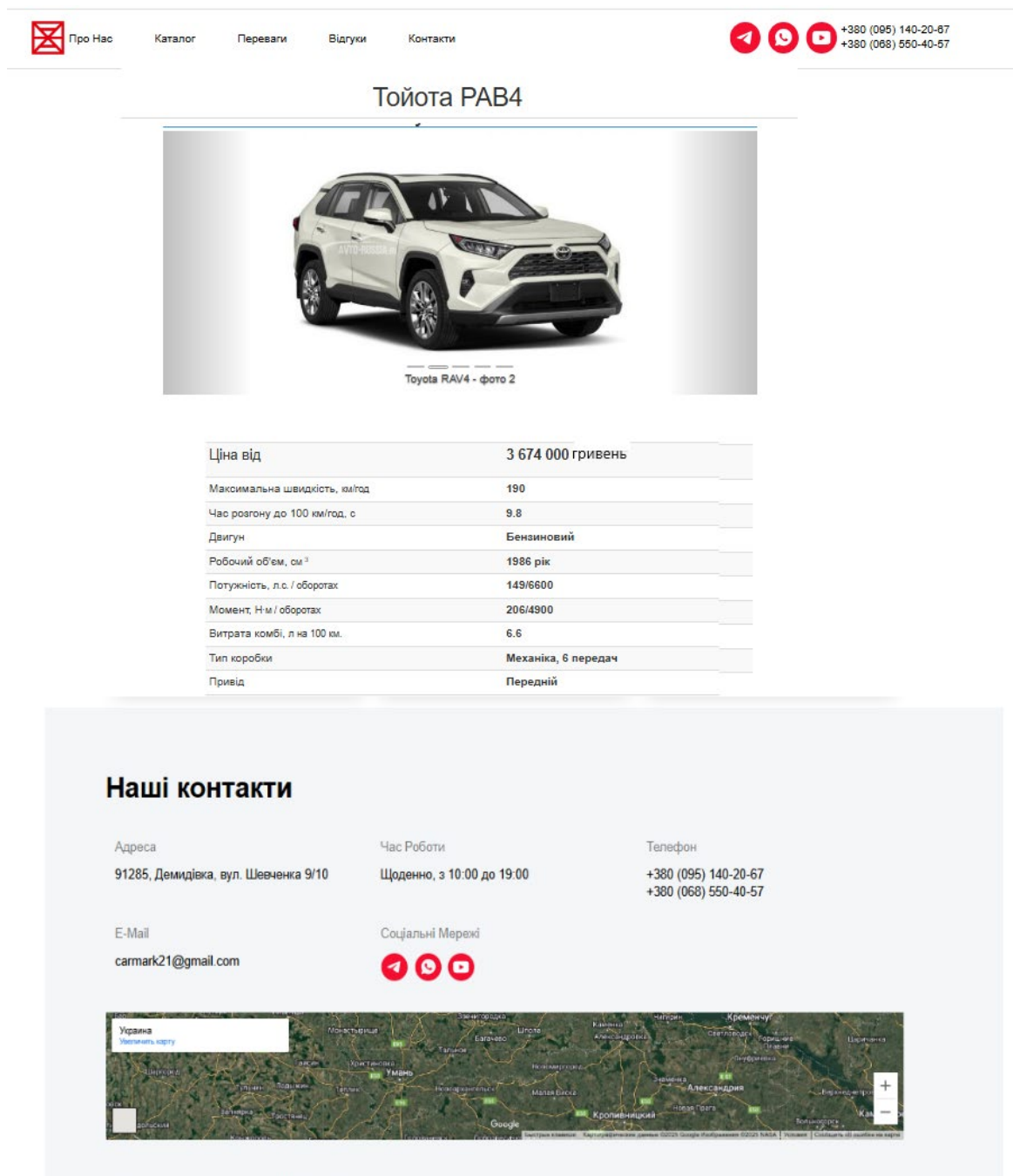


Рис. 4.9 – Сторінки catalog.html, model.html

4.4 Створення структури та опис React-компонентів панелі адміністратора

Згідно з усіма правилами побудови React-додатків, вся програма має бути розбита на низку окремих взаємопов'язаних між собою компонентів, які, у свою чергу, і формують структуру програми. Виходячи з цієї концепції, а також

ґрунтуючись на результатах проектування та інших вимог, можна скласти певну ієрархію компонентів. У загальному вигляді попередня структура виглядає так:

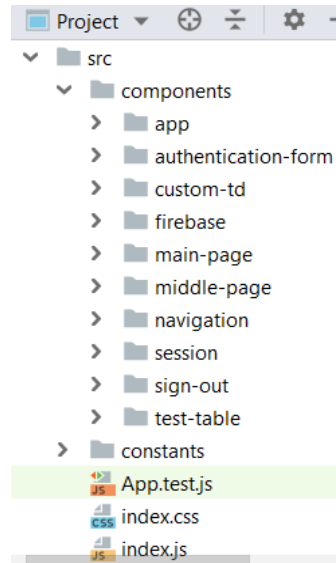


Рис. 4.10 – Структура програми

У кореневому файлі `index.js` відображається лише імпортований компонент-менеджер `app.js`, який взаємопов'язаний з усіма іншими і є керуючим[9].

Кожен компонент представлений у вигляді папки, в якій знаходяться 2 однойменні файли з розширеннями `.js` і `.css`. У першому міститься код компонента, а другий несе у собі необхідні стилі для забезпечення дизайнерської реалізації його зовнішнього вигляду, на думку розробника. І завершує послідовність файл із розширенням `.js` - `index.js`. Справа в тому, що у `webpack` є механізм, який полягає в наступному: якщо ми імпортуємо не файл, а папку, то всередині цієї папки веб-пакет спробує знайти файл, який називається `index.js`. За його наявності здійснюється імпорт за промовчанням. Це робиться для зручнішого запису адреси файлу, який потрібно імпортувати[9; 14]

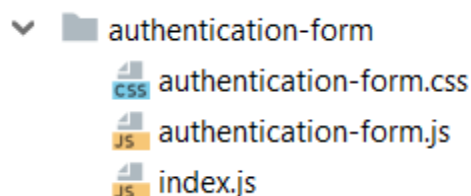


Рис. 4.11 – Список файлів в одній із папок структури

4.5 Забезпечення зв'язку між компонентами шляхом використання стандартної бібліотеки маршрутизації React - React Router

Сполучною ланкою між усіма компонентами виступає інструмент React Router. Він також забезпечує їх виклик як реакцію на будь-яку подію. Для встановлення потрібно прописати в терміналі команду `npm-install react-router-dom`.

Роутинг UI-додатків досить складний, тому краще його розбити на окремі блоки, так звані сторінки. випадку, всі «сторінки» вимагають ідентифікації. Як ідентифікатор може бути URL-подібна структура[26].

```
export const START_PAGE = '/';  
export const SIGN_IN = '/signin';  
export const MIDDLE_PAGE = '/mainpage';  
export const TEST = '/ Test';
```

Крім механізму включення та вимкнення певних компонентів, бібліотека React Router оновлює поточну URL-адресу, щоб новий URL вказував саме на вибрану сторінку, Route – складова частина Router - порівнює поточну адресу і значення path, а потім вирішує, чи слід прямо зараз відобразити вміст чи ні. Route-ами. Перевірку на точний збіг шляху реалізує параметр `exact`. В окремих випадках, для реалізації складної навігації, доцільно використовувати функцію-компонент вищого порядку з `route`, здатним передавати в інший компонент об'єкти реактивного router: `history`, `location` і `match`. Перший об'єкт додає новий елемент до історії браузера. Останній компонент буде використовуватися під час роботи з базою після подальшої інтеграції з сервісом Firebase.

```
<div> <AuthUserContext.Provider value={this.state.authUser}> <Router> <Route exact  
path={ROUTES.START_PAGE} component={MiddlePage} /> <Route exact  
path={ROUTES.SIGN_IN} component={Form} /> <Navigation aut exact  
path={ROUTES.MIDDLE_PAGE} component={Table} /> <Route exact path={ROUTES.TEST}  
component={MainPage} /> </Router> </AuthUserContext.Provider></div>
```

4.6 Конфігурування складових компонентів

Як уже було зазначено в розділі 3.1, додаток буде складатися з трьох основних компонентів, а також окремої технічної частини, що налаштовується. Найпростішим буде так званий стартовий компонент, куди користувач потрапляє відразу після запуску програми в браузері[4; 14]; .

На окрему увагу заслуговує форма авторизації – наступний за ієрархією та складністю компонент. Саме художнє оформлення форми, поля та правила заповнення є стандартними. Проте є деякі особливості. Насамперед – це наявність перевірки від Google – reCAPTCHA для React. Вона встановлюється за допомогою команди npm і react-recaptcha. reCAPTCHA - це безкоштовний сервіс, який захищає сайт або веб-орієнтовану програму від спаму та зловживань. Він використовує вдосконалений механізм аналізу ризиків, щоб відрізнити людей і ботів - поставляється у вигляді віджету, який ви можете легко додати до свого блогу, форуму, реєстраційної форми тощо.

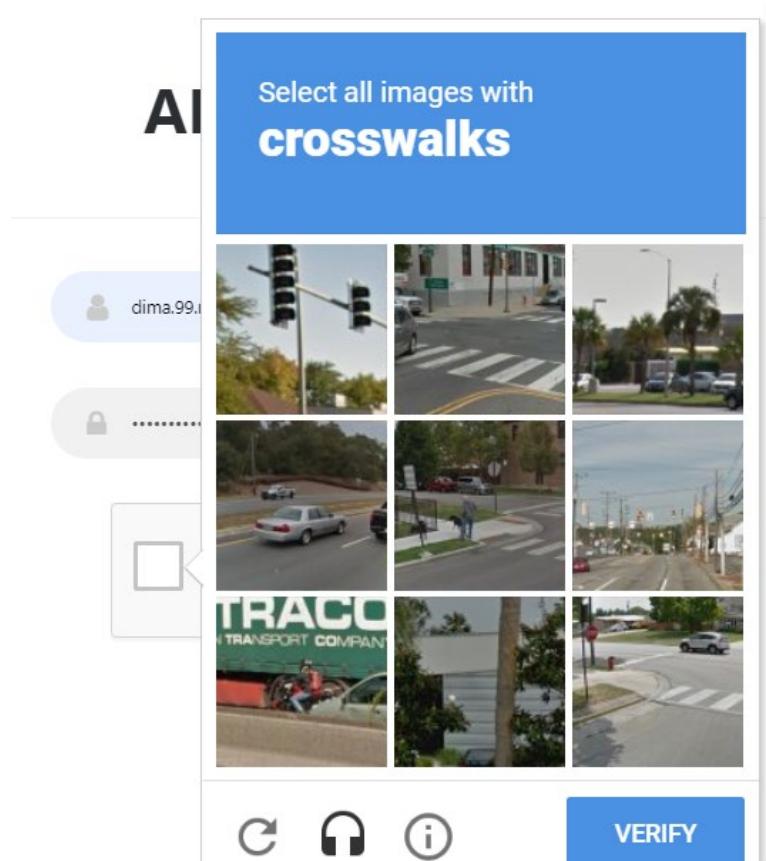


Рис. 4.12 –ReCAPTCHA у роботі

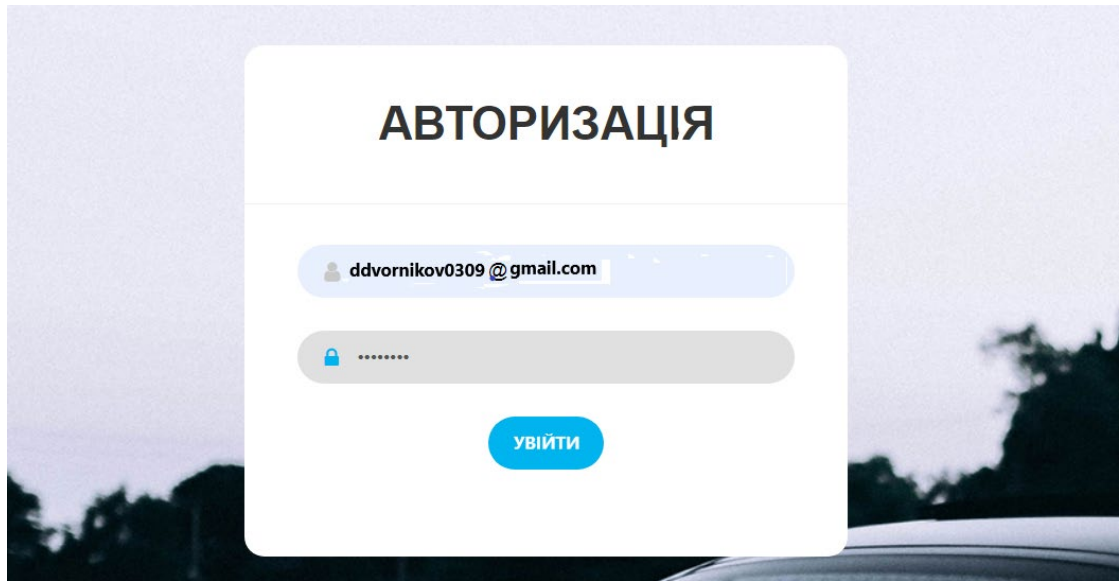


Рис. 4.13 – Форма авторизації

Також зобразимо код клієнтське частину, її код наведений нижче. Вона містить у собі можливості для редагування полів таблиць та відправляти результат на сервер через форму:

```

1  <script type="text/javascript">
2      let gk_isXlsx = false;
3      let gk_xlsxFileLookup = {};
4      let gk_fileData = {};
5      function filledCell(cell) {
6          return cell !== '' && cell !== null;
7      }
8      function loadFileData(filename) {
9          if (gk_isXlsx && gk_xlsxFileLookup[filename]) {
10             try {
11                 const workbook = XLSX.read(gk_fileData[filename], { type: 'base64' });
12                 const firstSheetName = workbook.SheetNames[0];
13                 const worksheet = workbook.Sheets[firstSheetName];
14
15                 // Convert sheet to JSON to filter blank rows
16                 let jsonData = XLSX.utils.sheet_to_json(worksheet, { header: 1, blankrows: false, defval: '' });
17                 // Filter out blank rows (rows where all cells are empty, null, or undefined)
18                 let filteredData = jsonData.filter(row => row.some(filledCell));
19
20                 // Heuristic to find the header row by ignoring rows with fewer filled cells than the next row
21                 let headerRowIndex = filteredData.findIndex((row, index) =>
22                     row.filter(filledCell).length >= filteredData[index + 1].filter(filledCell).length
23                 );
24                 // Fallback
25                 if (headerRowIndex === -1 || headerRowIndex > 25) {
26                     headerRowIndex = 0;
27                 }
28
29                 // Convert filtered JSON back to CSV
30                 let csv = XLSX.utils.aoa_to_sheet(filteredData.slice(headerRowIndex)); // Create a new sheet from filtered array
31                 csv = XLSX.utils.sheet_to_csv(csv, { header: 1 });
32                 return csv;
            }
          }
        }
      }

```

```

33         } catch (e) {
34             console.error(e);
35             return "";
36         }
37     }
38     return gk_fileData[filename] || "";
39 }
40 </script>`html
41 <!DOCTYPE html>
42 <html lang="uk">
43 <head>
44 <meta charset="UTF-8">
45 <meta name="viewport" content="width=device-width, initial-scale=1.0">
46 <title>Адмін-панель</title>
47 <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
48 <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
49 <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
50 <script src="https://cdn.tailwindcss.com"></script>
51 <style>
52 .container { max-width: 1350px; margin: auto; padding: 0 15px; }
53 .section-title { font-weight: 900; font-size: 36px; margin-bottom: 75px; }
54 .btn-default { background-color: #F50E30; color: white; padding: 16px 55px; font-family: 'Open Sans', sans-serif; font-weight: 700; }
55 .btn-default:hover { background-color: #d00c27; }
56 </style>
57 </head>
58 <body>
59 <div id="root"></div>

```

Рис. 4.14 – Код клієнтської частини

Фронтенд створено на React із використанням CDN (React, ReactDOM, Babel) для простоти. Для стилізації використано Tailwind CSS через CDN, адаптоване до дизайну сайту[9; 27; .

Структура адмін-панелі

Адмін-панель має вкладки для кожної таблиці. Кожна вкладка містить:

- Таблицю із даними (з можливістю редагування та видалення).
- Форму для додавання/оновлення записів із випадаючими списками для зовнішніх ключів (напр., id_brend, id_model).
- Кнопки у стилі .btn-default.

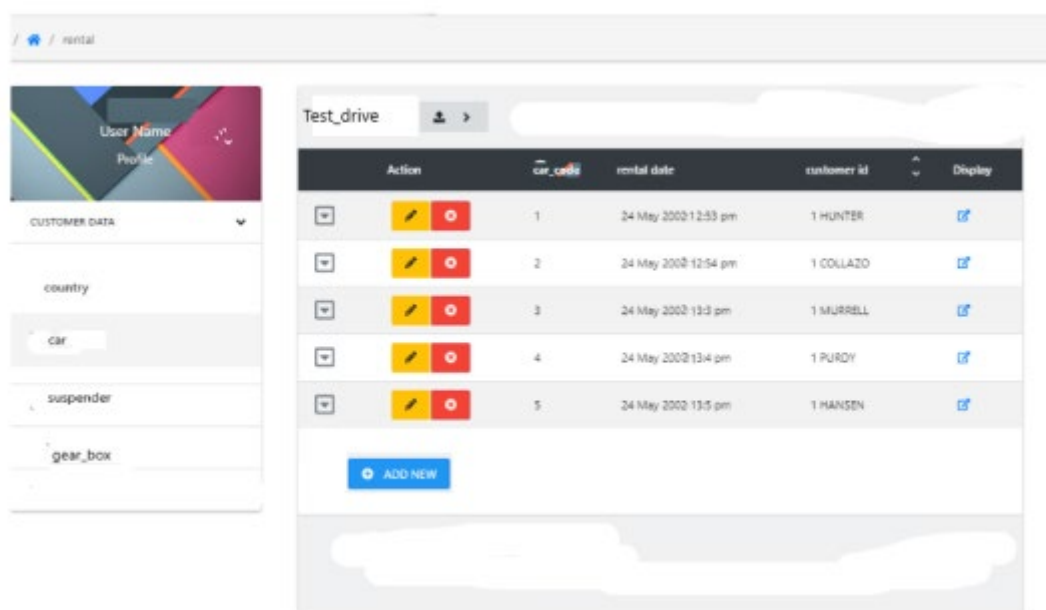


Рис. 4.15 – Вигляд адмін панелі (таблиця test_drive)

Також серверна частина, що обробляє запити, вносить зміни, якщо вони є, до бази даних та відправляє код відповіді, наприклад, до таблиці моделі:

```

197
198
199 useEffect(() => {
200   const fetchRelatedData = async () => {
201     const endpoints = ['brands', 'models', 'cars', 'carcase_types', 'drive_units', 'engines', 'gearboxes', 'suspenders'];
202     const related = {};
203     for (const endpoint of endpoints) {
204       try {
205         const response = await fetch(`http://localhost:5001/api/${endpoint}`);
206         related[endpoint] = await response.json();
207       } catch (error) {
208         console.error(`Error fetching ${endpoint}:`, error);
209       }
210     }
211     setRelatedData(related);
212   };
213   fetchRelatedData();
214 }, []);
215
216 const handleSubmit = async (e) => {
217   e.preventDefault();
218   const endpoint = tabs.find(t => t.id === tab).endpoint;
219   const method = editId ? 'PUT' : 'POST';
220   const url = editId ? `http://localhost:5001/api/${endpoint}/${editId}` : `http://localhost:5001/api/${endpoint}`;
221   try {

```

```

267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293

```

```

<form onSubmit={handleSubmit} className="mb-8">
  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
    {currentTab.fields.map(field => (
      <div key={field.name}>
        <label className="block mb-2">{field.label}</label>
        {field.type === 'select' ? (
          <select
            value={form[field.name] || ''}
            onChange={e => setForm({ ...form, [field.name]: e.target.value })}
            className="w-full p-2 border rounded"
          >
            <option value="">Виберіть</option>
            {(field.options || []).map(option => (
              <option key={option[field.key] || option.value} value={option[field.key] || option.value}>
                {option[field.value] || option.label}
              </option>
            ))}
          </select>
        ) : field.type === 'textarea' ? (
          <textarea
            value={form[field.name] || ''}
            onChange={e => setForm({ ...form, [field.name]: e.target.value })}
            className="w-full p-2 border rounded"
            rows="4"
          ></textarea>
        ) : (
          <input

```

Рис. 4.16 – Код серверної частини

Висновок по розділу 4

Спроектовано та розроблено веб-орієнтовану систему підтримки діяльності автосалону.

Описано основні React-компоненти, налагоджено забезпечення взаємодії між ними за допомогою бібліотеки React Routing, створено сайт, реалізована серверна частина через Node.js, а також реалізована та під'єднана база даних через СУБД MySQL.

ВИСНОВОК

Результатом виконаної бакалаврської роботи стала розроблена веб-орієнтована системи для підтримки діяльності автосалону. Першим етапом ми саме проаналізували функціонали вже існуючих та впроваджених систем та сайтів, що дозволяють виконувати аналогічну роботу, згідно з вимогами до нашого проекту. До цих систем можна віднести великі інформаційні та провітницькі рішення, такі як Infocar, сайти-агрегатори оголошень avto.ria та переважна частина сайтів комерційних автосалонів. Таким чином, на основі узагальнення зібраних даних, були сформовані вимоги до нашої системи, що поєднують в собі необхідний функціонал, дизайн та ефективність.

Стек технологій, визначений для розробки проекту ґрунтується на клієнт-серверній архітектурі та реалізовуватиметься через фреймворки та бібліотеки JavaScript, як на стороні фронтенду так і бекенду, зокрема react.js та node.js. Зберігання та обробку даних забезпечує реляційна БД на мові SQL, що відповідає третій нормальній формі. Необхідні пояснення про доцільність вибору саме описаних рішень детально обґрунтовуються другому розділі звіту.

Також, варто зазначити, що розмірковування по даній роботі та деякі рішення лягли в основу статті Розробка веб-орієнтованої системи для підтримки діяльності автосалону”.

Отже, за підсумками виконаної роботи та після неодноразових консультацій з керівником, побудована веб-орієнтована система, що повністю відповідає поставленим вимогам, усі технічні рішення реалізовані з використанням сучасних веб-технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Найкращі мови програмування 2020, які варто вивчати: веб-сайт. URL: <https://merehead.com/ru/blog/popular-programming-languages-2020/>
2. Javascript фреймворки. Що це таке? Навіщо? Чому? : веб-сайт. URL: <https://www.reclamare.ua/blog/javascript-frejmvorki/>
3. Гайна Г.А. Основи проектування баз даних. : Кондор, 2018. 208 с.
4. Мельник Р. А. Програмування веб-застосувань (фронт-енд та бек-енд) : Львівська політехніка, 2018. 246 с.
5. Джон Дакетт. HTML і CSS. Розробка та дизайн веб-сайтів. : К., 2013. 246 с.
6. Node.js + MySQL : веб-сайт. URL: https://www.w3schools.com/nodejs/nodejs_mysql.asp
7. CRM система для автосалонів та автодилерів : веб-сайт. URL: <https://wezom.com.ua/ua/blog/crm-sistema-dlja-avtosalonov-i-avtodilerov>
8. Дворніков Д., Козуб Ю. “Розробка веборієнтованої системи для підтримки діяльності автосалону”. Grail of Science, no. 52. 2025, pp. 686-691, doi:10.36074/grail-of-science.23.05.2025.093
9. Robert Wieruch. The Road to learn React – M-Publishing, 2025. - 260 с.
10. Алан Бол’є. Вивчаємо SQL : Науковий світ. 2023. 402 с.
11. Jonathan Wexler. Get programming with Node.js : Manning, 2019. 250 с.
12. MySQL documentation : веб-сайт. URL: <https://dev.mysql.com/doc/>
13. Що таке CRUD простими словами: функції, переваги та приклади : веб-сайт. URL: <https://highload.tech/uk/shho-take-crud-prostimi-slovami-funktsiyi-perevagi-ta-prikladi/>
14. Creating a React, Node, and Express App : веб-сайт. URL: <https://dev.to/techcheck/creating-a-react-node-and-express-app-1ieg>

15. flex - CSS: Cascading Style Sheets - MDN Web Docs : веб-сайт. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS/flex>.

16. CSS Flexbox Layout Guide : веб-сайт. URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>.

17. CSS Медіа запити - Приклади - W3Schools українською : веб-сайт. URL: https://w3schoolsua.github.io/css/css3_mediaqueries_ex.html

18. Jon Duckett. HTML and CSS: Design and Build Websites : Wiley, 2011. 512 с.

19. Що таке адаптивний дизайн сайту та як його зробити : веб-сайт. URL: <https://hostiq.ua/blog/ukr/adaptive-design/>

20. Що таке медіа запити CSS і для чого вони потрібні : веб-сайт. URL: <https://freehost.com.ua/ukr/faq/articles/chto-takoe-media-zaproshi-css-i-dlja-chego-oni-nuzhni/>.

21. Повний посібник (v1): Конструктор запитів : веб-сайт. URL: <https://yiiframework.com.ua/uk/doc/guide/database.query-builder/>

22. Мулеса О.Ю. Основи мови запитів SQL : посібник. Ужгород, 2015. 48 с. URL: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/8868/1/sql.pdf> (дата звернення: 06.05.2025)

23. MySQL Connector/Node.js : веб-сайт. URL: <https://dev.mysql.com/doc/dev/connector-nodejs/latest/>.

24. В.В. Біліченко, В.В. Варчук, О.В. Вдовиченко Менеджмент технічних служб на автотранспортних підприємствах : посібник. Вінниця: ВНТУ, 2006. 154 с. URL: <https://atm.vntu.edu.ua/subject/books/MTS%20na%20AT/Posibnyk.pdf> (дата звернення: 06.05.2025)

25. Anthony DeBarros Practical SQL: A Beginner's Guide to Storytelling with Data : No Starch Press, 2018. 312 с.

26. React Router Official Documentation: веб-сайт. URL: <https://reactrouter.com/>.

27. Порівняння популярних фреймворків для фронтенд-розробки: веб-сайт.
URL: <https://it-rating.ua/porivnyannya-populyarnih-freymvorkiv-dlya-frontend-rozrobkiu>

28. Postgres vs. MySQL: a Complete Comparison in 2025: веб-сайт. URL:
<https://www.bytebase.com/blog/postgres-vs-mysql/>.

29. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management System: веб-сайт. URL:
<https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>.

30. MySQL vs SQL Server: Your Ultimate Comparison Guide (2025) : веб-сайт.
URL: <https://www.astera.com/knowledge-center/mysql-sql-server/>.

31. SQL vs NoSQL: 5 Critical Differences: веб-сайт.
URL: <https://www.integrate.io/blog/the-sql-vs-nosql-difference>.

32. Difference between SQL and NoSQL: веб-сайт. URL:
<https://www.geeksforgeeks.org/difference-between-sql-and-nosql/>.

ДОДАТОК А

	GS 230525-086 dated 23.05.2025	
<h1>CERTIFICATE</h1> <h2>OF PARTICIPATION AND PUBLICATION</h2>		
<h3>Dmytro Duornikov</h3>		
		
<p>participated in the IV Correspondence International Scientific and Practical Conference</p> <p>Open science nowadays: main mission, trends and instruments, path and its development</p> <p>held on May 23th, 2025 by</p> <p>NGO European Scientific Platform (Vinnytsia, Ukraine) LLC International Centre Corporate Management (Vienna, Austria)</p>		
<p>and published scientific paper</p> <p>РОЗРОБКА ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ АВТОСАЛОНУ</p>		
<p>In Periodical scientific journal «GRAIL OF SCIENCE»</p> <p>№ 52 ISSN 2710-3056; Media identifier R30-02704; DOI 10.36074/grail-of-science.23.05.2025</p>		
<div>0.6 ECTS credits (18 hours) Recommended by the Academic Council of the «Institute of Scientific and Technical Integration and Cooperation». Protocol № 20 from May 22th, 2025.</div>		
<p>Head of the NGO «European Scientific Platform» Chairman of the Organizing committee GOLDENBLAT MIRIAM</p>	<p>Head of Community Outreach of the LLC «International Centre Corporate Management» RACHAEL APARO</p>	   
		

ДОДАТОК Б

```
1 <script type="text/javascript">
2   let gk_isXlsx = false;
3   let gk_xlsxFileLookup = {};
4   let gk_fileData = {};
5   function filledCell(cell) {
6     return cell !== '' && cell !== null;
7   }
8   function loadFileData(filename) {
9     if (gk_isXlsx && gk_xlsxFileLookup[filename]) {
10       try {
11         const workbook = XLSX.read(gk_fileData[filename], { type: 'base64' });
12         const firstSheetName = workbook.SheetNames[0];
13         const worksheet = workbook.Sheets[firstSheetName];
14
15         // Convert sheet to JSON to filter blank rows
16         let jsonData = XLSX.utils.sheet_to_json(worksheet, { header: 1, blankrows: false, defval: '' });
17         // Filter out blank rows (rows where all cells are empty, null, or undefined)
18         let filteredData = jsonData.filter(row => row.some(filledCell));
19
20         // Heuristic to find the header row by ignoring rows with fewer filled cells than the next row
21         let headerRowIndex = filteredData.findIndex((row, index) =>
22           row.filter(filledCell).length >= filteredData[index + 1]?.filter(filledCell).length
23         );
24         // Fallback
25         if (headerRowIndex === -1 || headerRowIndex > 25) {
26           headerRowIndex = 0;
27         }
28
29         // Convert filtered JSON back to CSV
30         let csv = XLSX.utils.aoa_to_sheet(filteredData.slice(headerRowIndex)); // Create a new sheet from filtered array
31         csv = XLSX.utils.sheet_to_csv(csv, { header: 1 });
32         return csv;
33       } catch (e) {
34         console.error(e);
35         return "";
36       }
37     }
38     return gk_fileData[filename] || "";
39   }
40 </script>`html
41 <!DOCTYPE html>
42 <html lang="uk">
43 <head>
44   <meta charset="UTF-8">
45   <meta name="viewport" content="width=device-width, initial-scale=1.0">
46   <title>Адмін-панель</title>
47   <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
48   <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
49   <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
50   <script src="https://cdn.tailwindcss.com"></script>
51   <style>
52     .container { max-width: 1350px; margin: auto; padding: 0 15px; }
53     .section-title { font-weight: 900; font-size: 36px; margin-bottom: 75px; }
54     .btn-default { background-color: #F50E30; color: white; padding: 16px 55px; font-family: 'Open Sans', sans-serif; font-weight: 700; }
55     .btn-default:hover { background-color: #d00c27; }
56   </style>
57 </head>
58 <body>
59   <div id="root"></div>
```

```

58 </body>
59 <div id="root"></div>
60 <script type="text/babel">
61   const { useState, useEffect } = React;
62
63   const AdminPanel = () => {
64     const [tab, setTab] = useState('brands');
65     const [data, setData] = useState({});
66     const [form, setForm] = useState({});
67     const [editId, setEditId] = useState(null);
68     const [relatedData, setRelatedData] = useState({});
69
70     const tabs = [
71       { id: 'brands', name: 'Бренди', endpoint: 'brands', fields: [
72         { name: 'brend_name', label: 'Назва бренду', type: 'text' },
73         { name: 'brend_name_rus', label: 'Назва бренду (укр)', type: 'text' },
74         { name: 'logo', label: 'Логотип (URL)', type: 'text' } ] },
75       { id: 'models', name: 'Моделі', endpoint: 'models', fields: [
76         { name: 'model_name', label: 'Назва моделі', type: 'text' },
77         { name: 'id_brend', label: 'Бренд', type: 'select', options: relatedData.brands, key: 'id_brend', value: 'brend' },
78         { name: 'main_view_link', label: 'Основне фото (URL)', type: 'text' },
79         { name: 'id_country', label: 'Країна', type: 'select', options: relatedData.countries, key: 'id_country', value: 'country' },
80         { name: 'model_year', label: 'Рік випуску', type: 'number' },
81         { name: 'price', label: 'Ціна', type: 'number' },
82         { name: 'description', label: 'Опис', type: 'textarea' } ] },
83       { id: 'cars', name: 'Характеристики авто', endpoint: 'cars', fields: [
84         { name: 'id_model', label: 'Модель', type: 'select', options: relatedData.models, key: 'id_model', value: 'model' },
85         { name: 'id_carcase', label: 'Тип кузова', type: 'select', options: relatedData.carcase_types, key: 'id_carcase', value: 'carcase' },
86         { name: 'car_length', label: 'Довжина', type: 'number' },
87         { name: 'car_width', label: 'Ширина', type: 'number' } ] } ] },
88     display: ['brend_name_rus', 'brend_name', 'logo'],
89     display: ['model_name', 'brend_name_rus', 'c_name', 'model_year', 'price'],

```

```

197   useEffect(() => {
198     const fetchRelatedData = async () => {
199       const endpoints = ['brands', 'models', 'cars', 'carcase_types', 'drive_units', 'engines', 'gearboxes', 'suspensers'];
200       const related = {};
201       for (const endpoint of endpoints) {
202         try {
203           const response = await fetch(`http://localhost:5001/api/${endpoint}`);
204           related[endpoint] = await response.json();
205         } catch (error) {
206           console.error(`Error fetching ${endpoint}:`, error);
207         }
208       }
209       setRelatedData(related);
210     };
211     fetchRelatedData();
212   }, []);
213
214   const handleSubmit = async (e) => {
215     e.preventDefault();
216     const endpoint = tabs.find(t => t.id === tab).endpoint;
217     const method = editId ? 'PUT' : 'POST';
218     const url = editId ? `http://localhost:5001/api/${endpoint}/${editId}` : `http://localhost:5001/api/${endpoint}`;
219     try {
220

```

```

267   <form onSubmit={handleSubmit} className="mb-8">
268     <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
269       {currentTab.fields.map(field => {
270         <div key={field.name}>
271           <label className="block mb-2">{field.label}</label>
272           {field.type === 'select' ? (
273             <select
274               value={form[field.name] || ''}
275               onChange={e => setForm({ ...form, [field.name]: e.target.value })}
276               className="w-full p-2 border rounded"
277             >
278               <option value="">Вибрати</option>
279               {(field.options || []).map(option => {
280                 <option key={option[field.key] || option.value} value={option[field.key] || option.value}>
281                   {option[field.value] || option.label}
282                 </option>
283               })}
284             </select>
285           ) : field.type === 'textarea' ? (
286             <textarea
287               value={form[field.name] || ''}
288               onChange={e => setForm({ ...form, [field.name]: e.target.value })}
289               className="w-full p-2 border rounded"
290               rows="4"
291             ></textarea>
292           ) : (
293             <input

```

ДОДАТОК В

```
1  ``javascript
2  const express = require('express');
3  const mysql = require('mysql2/promise');
4  const cors = require('cors');
5  const app = express();
6
7  app.use(cors());
8  app.use(express.json());
9
10 // Підключення до MySQL
11 const pool = mysql.createPool({
12   host: 'localhost',
13   user: 'root',
14   password: '0844Dv_1999',
15   database: 'autohouse'
16 });
17
18 // Універсальні функції
19 const getAll = async (table, res, joinQuery = `SELECT * FROM ${table}`) => {
20   try {
21     const [rows] = await pool.query(joinQuery);
22     res.json(rows);
23   } catch (error) {
24     res.status(500).json({ error: error.message });
25   }
26 };
27
28 const addRecord = async (table, fields, values, res) => {
29   try {
30     await pool.query(`INSERT INTO ${table} (${fields.join(', ')}) VALUES (${fields.map(() => '?').join(', ')}, values);`);
31     res.status(201).json({ message: `${table} додано` });
32   } catch (error) {
33     res.status(500).json({ error: error.message });
34   }
35 };
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64 app.put('/api/brands/:id', async (req, res) => {
65   const { brand_name, brand_name_rus, logo } = req.body;
66   updateRecord('car_brand', 'id_brand', req.params.id, ['brand_name', 'brand_name_rus', 'logo'],
67     [brand_name, brand_name_rus, logo], res);
68 });
69 app.delete('/api/brands/:id', async (req, res) => deleteRecord('car_brand', 'id_brand', req.params.id, res));
70
71 // Моделі
72 app.get('/api/models', async (req, res) => {
73   getAll('model', res, `
74     SELECT m.*, b.brand_name_rus, c.c_name
75     FROM model m
76     JOIN car_brand b ON m.id_brand = b.id_brand
77     JOIN country c ON m.id_country = c.id_country
78   `);
79 });
80 app.post('/api/models', async (req, res) => {
81   const { model_name, id_brand, main_view_link, id_country, model_year, price, description } = req.body;
82   addRecord('model', ['model_name', 'id_brand', 'main_view_link', 'id_country', 'model_year', 'price', 'description'],
83     [model_name, id_brand, main_view_link, id_country, model_year, price, description], res);
84 });
85 app.put('/api/models/:id', async (req, res) => {
86   const { model_name, id_brand, main_view_link, id_country, model_year, price, description } = req.body;
87   updateRecord('model', 'id_model', req.params.id, ['model_name', 'id_brand', 'main_view_link', 'id_country', 'model_year',
88     [model_name, id_brand, main_view_link, id_country, model_year, price, description], res);
89 });
90 app.delete('/api/models/:id', async (req, res) => deleteRecord('model', 'id_model', req.params.id, res));
91
92 // Характеристики автомобілів
93 app.get('/api/cars', async (req, res) => {
94   getAll('car', res, `
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
```